Based on the original Microfiche, multiple pages appear to be missing from this document

HARDWARE SYNTHESIS FROM DDL DESCRIPTION

(NASA-CR-161667) HARDWARE SYNTHESIS FROM DDL DESCRIPTION Annual Technical Report, 1 Oct. 1979 - 30 Sep. 1980 (Alabama Univ. in Huntsville.) 181 p HC A09/MF A01 CSCL 09B
N81-19778
Unclas
G3/60 41689

Prepared by

SAJJAN G. SHIVA and ANIL M. SHAH
Computer Science Department
The University of Alabama in Huntsville
Huntsville, Alabama 35807

Second Annual Technical Report
October 1980

for

NAS8 - 33096
DESIGN SYNTHESIS OF DIGITAL SYSTEMS
George C. Marshall Space Flight Center
Alabama, 35812

The University
Of Alabama
In Huntsville

FOREWORD

This is a technical summary of the research work conducted during October 1, 1979 to September 30, 1980 by The University of Alabama in Huntsville towards the fulfillment of the Contract NAS8-33096 from George C. Marshall Space Flight Center, Alabama. The NASA technical officer for this contract is Mr. Robert E. Jones.

ABSTRACT

The details of the digital systems can be conveniently input into the design automation system by means of Hardware Description Language (HDL). The Computer Aided Design and Test (CADAT) system at NASA MSFC is used for the LSI design. The Digital Design Language (DDL) has been selected as HDL for the CADAT System. DDL translator output can be used for the hardware implementation of the digital design. This Thesis addresses problems of selecting the standard cells from the CADAT standard cell library to realize the logic implied by the DDL description of the system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

The details of the digital systems design can be conveniently input into a design automation system by means of Hardware Description Languages (HDL). The use of HDLs in Large Scale Integrated (LSI) circuit design automation is not widespread for the following reasons [1]:

(1) The difficulty in translating the HDL description into an implementable format, such as the interconnection of standard circuit modules, logic diagram, etc..

(2) Non-uniform design methodologies: The designer in practice uses a variety of design techniques that can not be clearly identified as top-down or bottom-up methods.

(3) The time and cost involved in transporting and tailoring the HDL software developed at one design center to another.

The complexity of the LSI circuit is greatly increased due to the advent of Very Large Scale Integration (VLSI). Hence, the breadboard for a VLSI circuit is a VLSI circuit itself. A thorough verification of the VLSI design at the earliest stage in the design cycle is absolutely necessary to minimize the costs of mask fabrication and wafer processing brought about by the later changes in the design. The HDL can be used to provide such high-level verification capability of a VLSI design [1].

The Computer Aided Design And Test (CADAT) system of the National Aeronautics and Space Administration (NASA)/Marshall Space Flight Center (MSFC) is functionally organized as shown in Figure 1 [2]. The designer

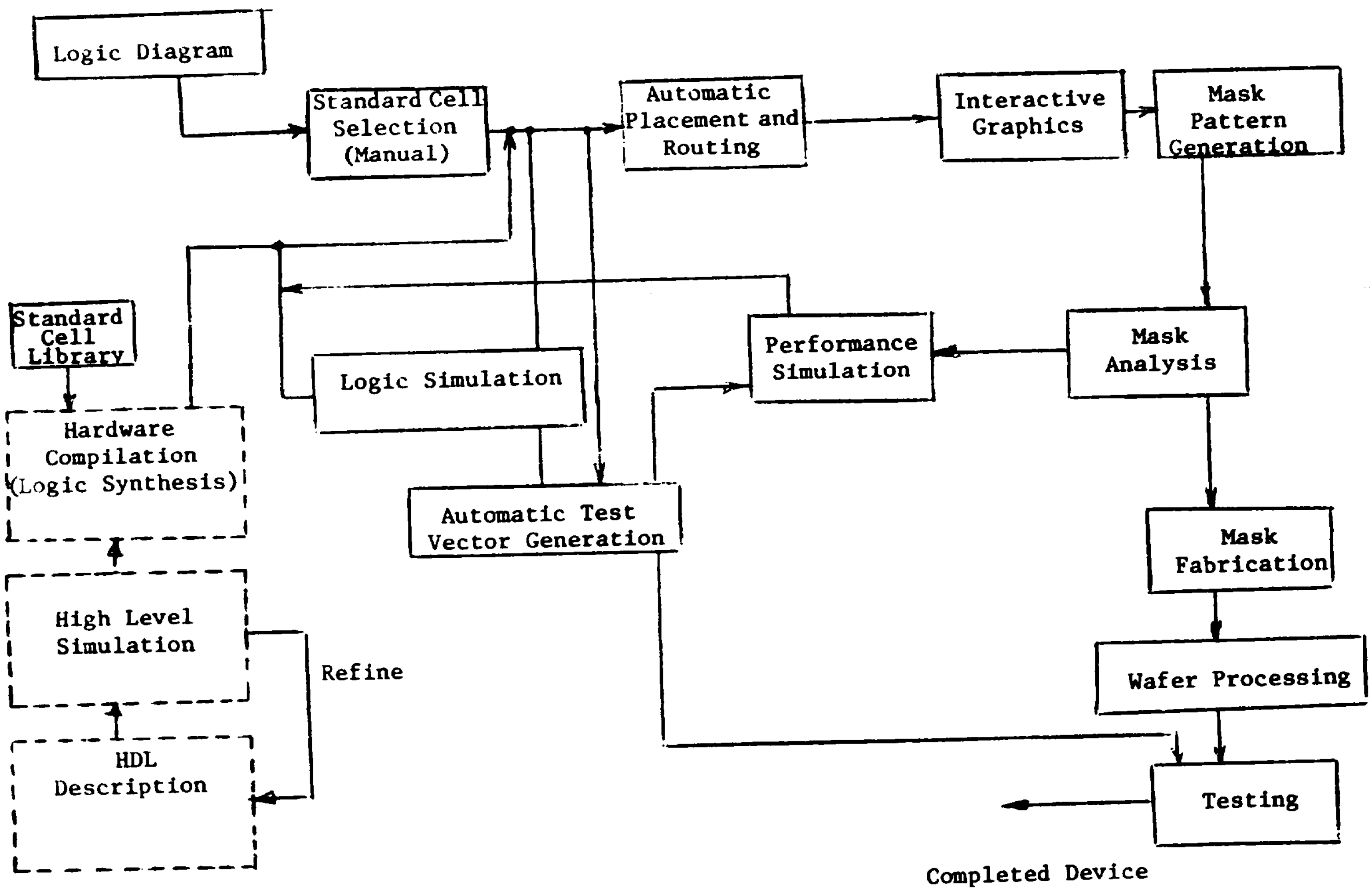


Figure 1: The LSI Design Process.

inputs the detail of the Integrated Circuit (IC) to CADAT as a set of standard cells and their interconnections. The standard cell selection is done manually from a standard cell library. This design description is at the logic diagram level. Detailed logic simulation and refinements are carried out on the design. The final design is input to the automatic test pattern generation and placement and routing programs. The IC mask pattern generation is refined interactively and mask analysis and performance simulation are done before fabricating the mask. The last two steps in the IC fabrication are the wafer processing and the final testing [2].

The HDL would help the designer to simulate his design and refine it at a high-level before entering his design into the current system. The logic synthesizer (hardware compiler) would allow automatic selection of the standard cells from the standard cell library and interconnection of these selected cells to realize the logic implied by the HDL description.

The Digital Systems Design Language (DDL) [3] has been selected [4] as the HDL for the CADAT system. The DDL Translator (DDLTRN) [3,4] generates a set of Boolean Equations (BE) and a set of Register Transfer Equations (RTE) from a DDL description of the digital system. This Report addresses the problem of selecting standard cells from the standard cell library and interconnecting these selected cells to realize the logic implied by the BEs and RTEs.

Chapter 2 is a brief review of literature on the Hardware Design Process, HDL and the CADAT system. Chapter 3 summarizes the constructs available in DDL and discusses the translator program DDLTRN. The Hardware Synthesis Algorithm (HSA) is described in Chapter 4. Chapter 5

provides the implementation details of HSA on SEL-32 computer system. Some design examples are provided in Chapter 6.

CHAPTER 2. HDLS IN HARDWARE DESIGN

This chapter provides a brief review of the Hardware Design methodologies. Uses of HDLs in the design process are identified along with requirements for HDL to be useful in the CADAT system. A brief description of the CADAT system is also given.

2.1 THE HARDWARE DESIGN PROCESS

It is difficult to describe the hardware design process formally, since it is still an art and depends to a large extent on the individual designer and on the specific design problem to be solved. Starting from a set of sometimes vague and incomplete specifications, the designer applies a series of successive improvements until the system can be realized within a given technological environment, or until it is clear that the specifications are not feasible [5].

It is impossible to create a final design at once unless the design is trivial. Two specific approaches to systems design are often recommended. The top-down approach is the approach where the designer divides a problem into a number of interconnected sub-problems. This process is then repeated until a solution to each of the sub-problems is known or until a well-known procedure can be applied to solve these sub-problems. In the bottom-up approach, several elementary compoenets are combined to form more complex ones until the complete system design is achieved. In practice, the designer uses a combination of both the approaches [6].

Formal methods do exist for solving certain problems, such as minimization of combinational circuits [7], or the state assignments for

sequential circuits [8]. However, a "Library" of examples is used by many designers to base their designs. These examples are drawn from the previous design experiences, the literature, or the class-room exposure. The advent of Medium Scale Integration (MSI) and LSI building blocks has magnified this tendency of constructing hardware design by "example" [5,6].

In the early stages of the design cycle more emphasis is placed on the behavior of the system than the structural aspects. In the later stages, the designer adds more and more structural information to the design in progress until the physical design can be implemented using the available components. The design of digital systems usually consists of several levels of refinements. In actual design practice, it is not desirable to predetermine the amount of detail at which a designer should work. The amount of detail is dependent on the technology used to implement a design. The amount of detail varies with the individual designer [6,9]. The levels in which a digital system can be described [2] are shown below:

(1) Algorithmic level which specifies only the algorithm used by the hardware for the problem solution,

(2) Processor Memory Switch (PMS) level which describes the system in terms of processing units, memory component, peripherals, and switching networks,

(3) Instruction level (programming level) where the instructions and their interpretation rules are specified,

(4) Register transfer level where the registers are system elements, and the data transfer between these registers are specified according to some rule,

(5) Switching circuit level where the structure consists of an interconnection of gates and flip-flops and the behavior is given by a set of BEs, and

(6) Circuit level where the gates and flip-flops are replaced by the circuit elements such as transistors, diodes, registers, etc..

2.2 COMPUTER HARDWARE DESCRIPTION LANGUAGE [2]

High Level Languages (HLL) are used by the software designers to express the algorithms in terms of language statements. On the other hand, HDLs are used by the digital circuit designers to describe the system being designed. The parallelism, nonrecursive nature, and the timing issues of the hardware can be easily described by an HDL.

The HDLs differ from the pure sequential nature of a general HLL. An HDL can also be classified as a procedural or a non-procedural language. Each statement in a non-procedural HDL description would contain a label which describes the condition under which the activities described by the statement are to be performed, and hence does not reflect the ordering of the activities. In a procedural HDL description, the statements are ordered. The activities described by the statements are performed in that order [2]. HDLs have been in existance since the early 1980's. However, within the last few years, a serious effort is being made to bring HDLs into the design process as useful tools. The major applications of an HDL [2,5] are listed below:

(1) Description of the behavior and/or structure of a system. This provides an accurate communication of the system details among designers and users.

(2) A convenient documentation tool to generate users manuals,

service manuals, etc..

(3) Input of the system description into a computer for simulation and design verification at various levels of details.

(4) Software generation tool at the preprototype levels, thus bridging the Hardware/Software development time gap.

(5) Convenient incorporation of design changes into the design documentation.

(6) Designer/user communication interface at the desired level of complexity.

(7) As the input to an automatic hardware synthesizer.

Several requirements can be imposed on an HDL to be suitable in an IC design automation environment. Some of such requirements are [2,10]:

(1) The language should serve as a medium at and below the register-transfer level of system description, since the majority of designs call for such a level of detail.

(2) A translator whose output is amenable to easier logic synthesis and test vector generation is required. A simulation capacity is required. The translator and simulator of HDL shall be written in a HLL for the portability aspect.

(3) The Language should be easy to learn and remember, since a hardware designer may not be a software expert.

(4) The design changes should be incorporated into the description and corresponding translation should be done preferably without a complete retranslation. This feature

will be useful for an interactive environment.

(5) The structural detail provided at any design level varies from designer to designer. The HDL should allow the designer to control the amount of the detail during each design phase.

(6) The description of the standard cells as system components should be straight-forward. If the system is partitioned by the designer to accomodate the standard cells, this partitioning should be retained by the HDL translator. This will enable a modular hardware synthesis.

The following five criteria were used in selecting a suitable HDL for the CADAT system at NASA-MSFC [2,10]:

(1) Activity

(2) Level of description

(3) Software availability and portability

(4) Ease of logic generation, and

(5) Modularity.

Activity criteria requires one to choose an available popular language. The process of improving and adding the newer capabilities to the selected HDL would be aided by the active interest of the other groups in the language. The level of description must be at the register transfer and/or below to aid the LSI design environment. The selected HDL should have a translator and a simulator. The general portability of the CADAT software should be maintained. Any HDL translator oriented towards providing information for a simulator, collects and rearanges the combinational logic and register transfers. Easier logic

generation must be possible from this intermediate translated description. The HDL description should be modular enough to reflect the modularity of the hardware, to enable easier understanding and modular design verification [2,10].

Among a field of more than 40 HDLs that are recently reported, only 4 prominent HDLs (Computer Design Language (CDL), A Hardware Programming Language (APHL), Digital Systems Design Language (DDL), and Instruction Set Processor (ISP)) passed activity criterion. These were further compared and analyzed with respect to the other criteria and the DDL was chosen [10,11], for the CADAT system.

2.3 CADAT SYSTEM [12,13]

The CADAT system of NASA-MSFC is used in the design and fabrication of ICs and evaluation of IC design in newer technologies. The CADAT system consists of 3 subsystems: Layout subsystem (LAYOUT), Evaluation subsystem (EVAL), and Design intent subsystem (LOGIC). The intended logic is input to logic, which generates the stimuli and expected performance data required by the EVAL and layout data for the LAYOUT. LOGIC consists of pre-processor (NTRAN), a post-processor (TPGITF), and a logic simulator (LOGSIM). The NTRAN accepts data in cell net format and provides circuit model data for the Test Pattern Generation (TPG) and LOGSIM. The TPG post-processor (TPGITF) automatically supplies stimuli to the LOGSIM and The Field Effect Transistor logic simulator (FETLOG). LOGSIM outputs the expected performance of the intended design to the EVAL. EVAL also requires the mask geometries as input to provide the performance data. The first program of EVAL is the Mask Analysis Program (MAP), which looks at the geometric blocks on all mask levels and determines where transistors, feed-throughs, con-

ductors, guard-bonds, etc., exist. The MAP program redescribes the same mask pictures in terms of geometries tagged with their functions. These tagged geometries, together with definition of the process rules are checked for errors by MAP. MAP program also describes the circuitry on the masks in nodal format. This nodal data, together with the functionally tagged geometries are passed along to the Disjunct Analyzer (DJANAL) program. The DJANAL groups transistor circuits into the disjunct circuits for use by the FETLOG simulator. The stimuli from LOGIC is also input to the FETLOG. The performance evaluation flow is shown in Figure 2 [12].

The details of the design process discussed in Chapter 1 are shown in Figure 3 [2,13]. The DDL would help the designer to simulate his design and refine it at high level before entering his design into the current system. The Logic Synthesizer (DDLSYN) would allow automatic selection of the standard cells from the cell library and interconnection of these selected cells to realize the logic implied by the DDL description.

2.3.1 The Standard Cell Library for CADAT System

The standard cell library is an open-ended collection of logic circuits implemented with the Complementary Metal Oxide Semiconductor (C-MOS) technology. All standard cells have been defined, designed, topologically configured, analyzed, and then permanently stored for future use on magnetic tape. The present library is quite extensive and is designed to meet the present and anticipated C-MOS implementation needs. It is also possible to define and design new cells to meet unique future system requirements in the most efficient manner possible [13].

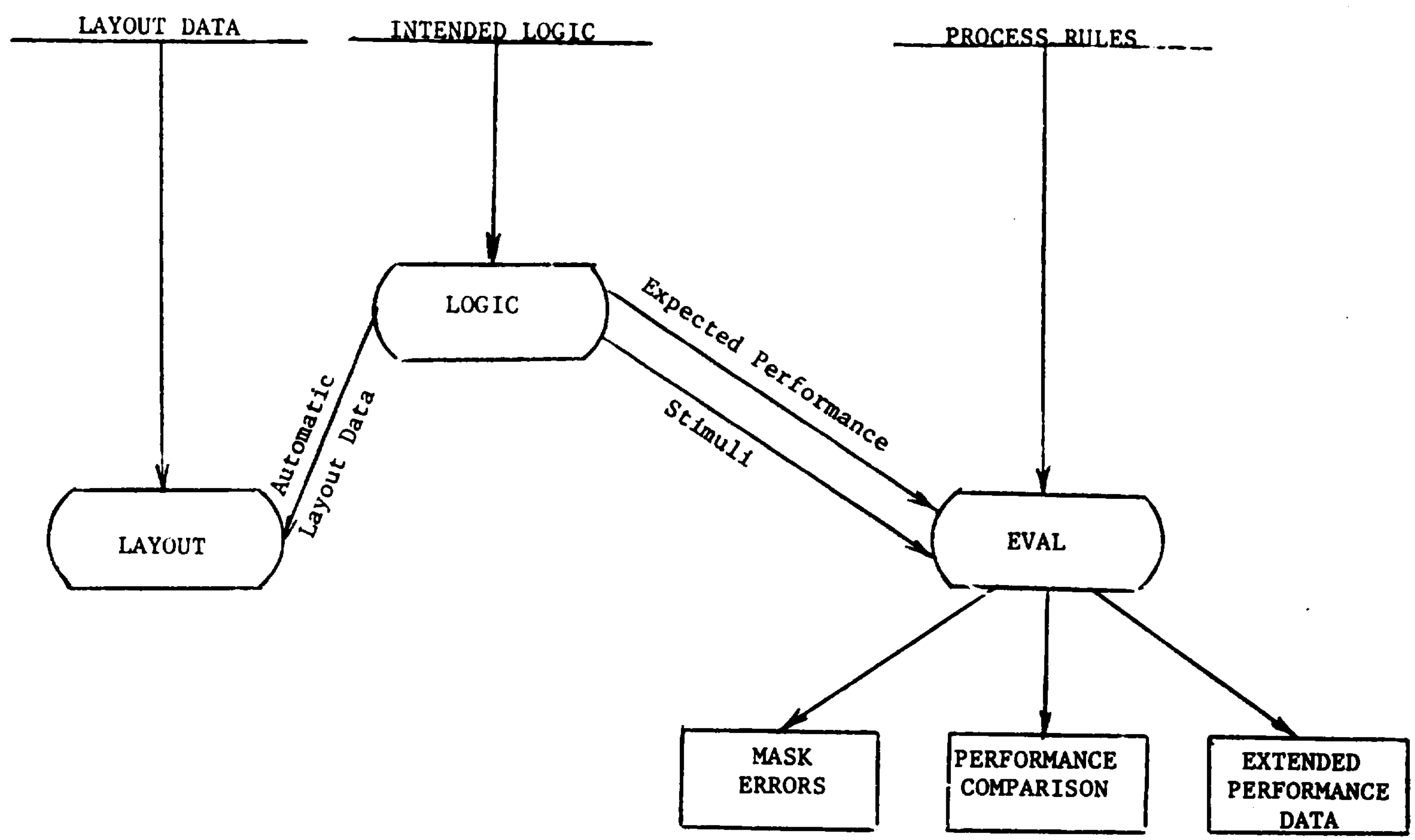


Figure 2: Performance Evaluation Flowchart.

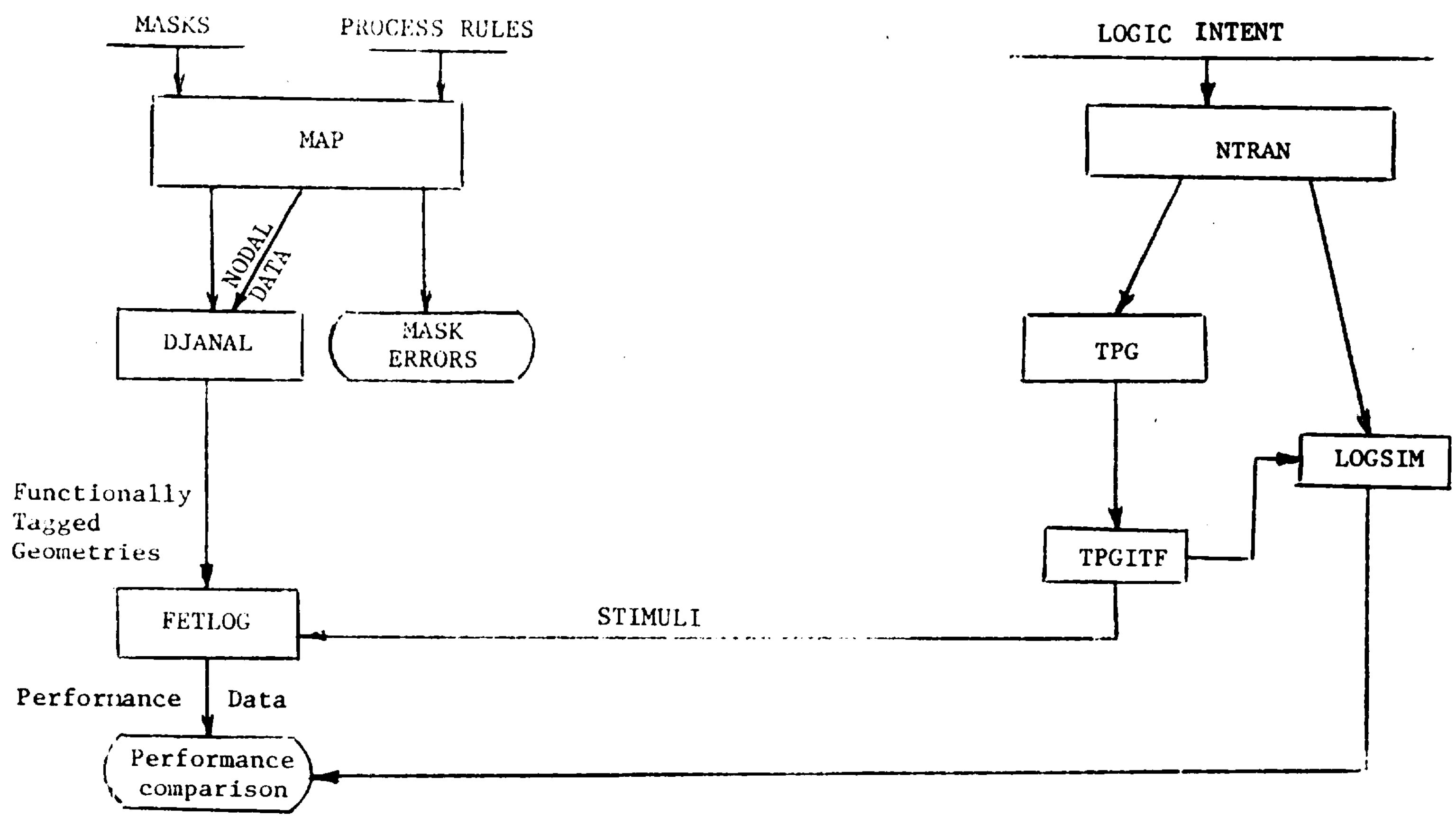


Figure 2: (Concluded)

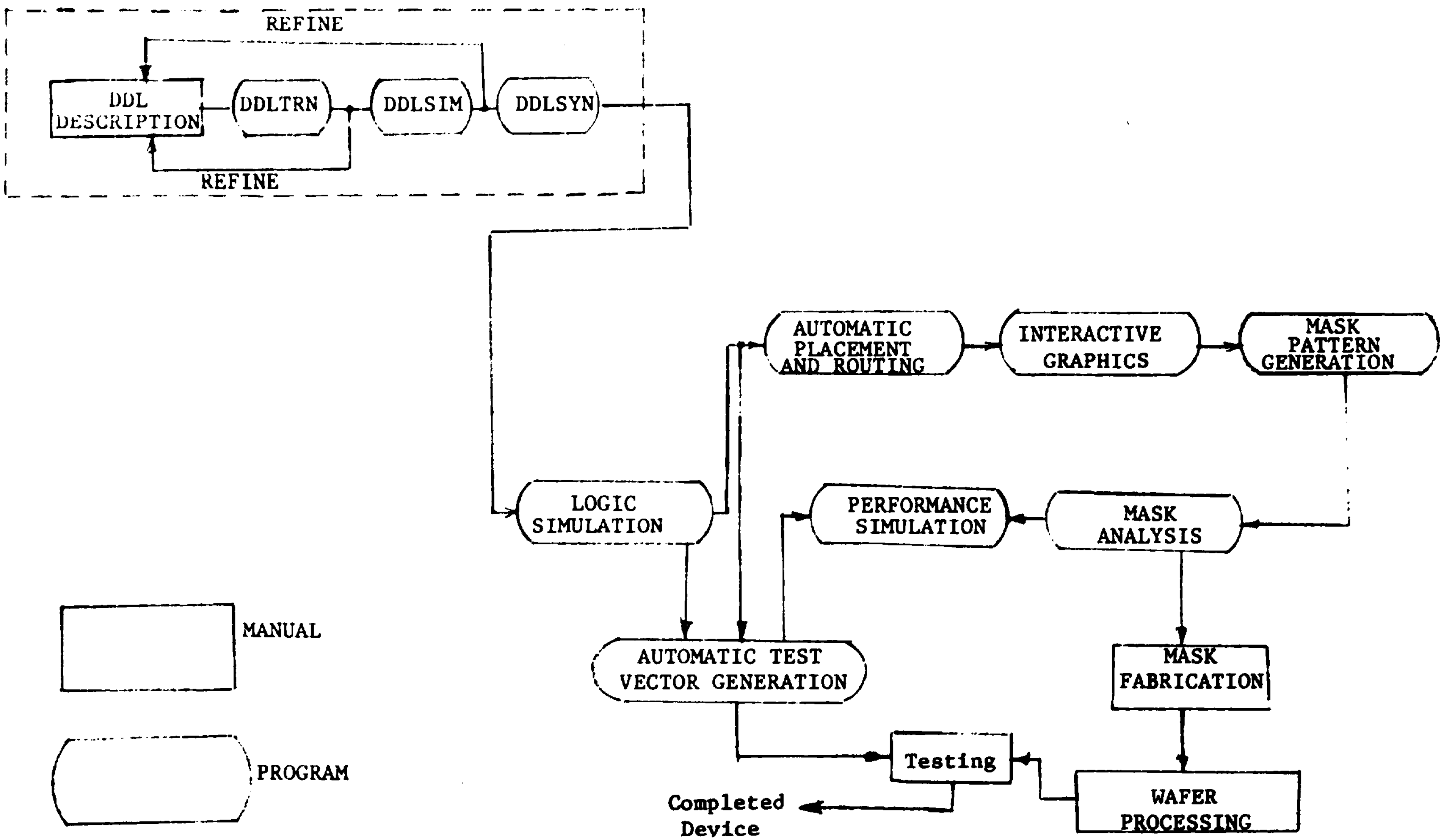


Figure 3: The CADAT System with DDL

Table 1 - CADAT Standard Cell Library (Partial)

Cell No.	Type	No. of Devices	Cell Width (mils)	Function	Literals/ Product Pattern (LPP)
1120	2 input NOR	4	5.8	$\overline{A+B}$	1,1
1130	3 input NOR	6	7.7	$\overline{A+B+C}$	1,1,1
1140	4 input NOR	8	9.6	$\overline{A+B+C+D}$	1,1,1,1
1220	2 input NAND	4	5.8	$\overline{A*F}$	2
1230	3 input NAND	6	7.7	$\overline{A*B*C}$	3
1240	4 input NAND	8	9.6	$\overline{A*B*C*D}$	4
1310	Buffer Inverter	2	3.9	$\overline{A}$	1
1300	Non-inverting Buffer	4	4.5	A	1
1620	2 input AND	6	5.8	A*B	2
1630	3 input AND	8	7.7	A*B*C	3
1640	4 input AND	10	9.6	A*B*C*D	4
1720	2 input OR	6	5.8	A+B	1,1
1730	3 input OR	8	7.7	A+B+C	1,1,1
1740	4 input OR	10	9.6	A+B+C+D	1,1,1,1
1800	4x2 input AND + 4 x NOR	16	17.2	$\overline{(AB+CD+EF+GH)}$	2,2,2,2
1860	2x2 input AND + 4 input NOR	12	13.7	$\overline{AB+E+F+CD}$	2,1,1,2
1870	2x2 input AND + 2 input NOR	8	9.6	$\overline{AB+CD}$	2,2
1890	3x2 input AND + 3 input NOR	12	16.9	$\overline{AB+CD+EF}$	2,2,2
2310	2 input EX-OR	8	7.8	A@B	-
1830	D-Flip-Flop	10	8.4	—	—

A partial list of the standard cells available in the CADAT system is given in Table 1. Each standard cell is defined by cell number as in Column 1. One of these standard cells can be used in the logic design by calling the desired cell by the cell number. Column 2 of Table 1 gives the type information for a cell. The number of devices for each cell and the cell width (as a measure of the silicon area needed) are shown in Column 3 and 4 respectively. Column 5 defines function realized by the cell. The last column shows the literals (LPP) in each product term of the function realized by the cell. The LPP with one product term (1,2,3,4) and those containing all 1's (11,111,1111) are single gate realizations. The four cells 2222,2112,222 and 22 realize more complex functions than a gate equivalent [10].

2.4 SUMMARY

A potentially important application of HDLs is as the input to a hardware compiler that automatically translates the high level language description into a logic design. This is extremely useful, since together with a register transfer level simulator, it would allow rapid and accurate hardware design.

CHAPTER 3. DIGITAL DESIGN LANGUAGE

DDL was first introduced in 1967 by Duley and Dietmeyer [3,14]. A Fortran based implementation was done at the University of Wisconsin [15,16]. An Algol based version was implemented by Duley at Hewlett Packard. A PASCAL version based on the Algol version was implemented at Stanford University [5]. The translator (DDLTRN) and Simulator (DDLSIM) are implemented in FORTRAN at NASA-MSFC [4]. The DDLTRN translates a DDL description into a set of BEs and RTEs. The simulator program enables the designer to verify his design at the register transfer level. The output of the DDLTRN is an input to the simulator. The simulator commands enable the designer to control the simulation and input/output data during the simulation.

The language details will be described first. Brief discussion of DDLTRN Data Structure and six phases of translation will be described next, followed by an outline of the DDLSIM.

3.1 THE LANGUAGE [3,4,14]

DDL is a block oriented language. The blocks of a description correspond to the subsystems, parts, etc. of the hardware system [15] being described. For the language details of DDL refer to [3,4,14]. The syntax diagrams for DDL declarations are given in Appendix A. The hierarchy of DDL description is shown in Figure 4. In DDL, the structural facilities (e.g. Register, Terminal, etc.) are explicitly declared. The logical statements can be formed using the available primitive operators (e.g. AND, OR, etc.). The functional specification contains

SY

GLOBAL FACILITIES

OPERATOR$_1$ ← Global Operators

OPERATOR$_2$

⋮

OPERATOR$_n$

AUTOMATION$_1$

LOCAL FACILITIES

OPERATOR$_1$ ← Local Operators

OPERATOR$_2$

⋮

STATE DECLARATION

STATE$_1$

STATE$_2$

⋮

AUTOMATION$_2$

⋮

AUTOMATION$_3$

⋮

ENDSY

Figure 4: Hierarchy of DDL Description

hardware components for performing logical and arithmetic functions such as adders, counters, etc.

One of the major characteristics of the DDL is its assumption that the digital system is described as a finite state machine. Both data handling and control facilities are found in an automaton. Each automaton of a system is considered to be a finite state machine which exerts control over data facilities. The timing mode of an automaton may be synchronous or asynchrouous [14].

If the flow of information to or from a facility is controlled entirely by a single automaton, then that facility is "Local" and considered to be a part of the controlling automaton. If two or more automata exert control over a facility, it is a "Global" facility [14]. All communications between automata will take place via global facilities as shown in Figure 5.

3.2 DDL TRANSLATOR (DDLTRN) [4,15]

The DDLTRN converts the DDL description of the digital system into a Facility table, a set of BE and a set of Conditioned RTE. The facility table is simply a list of the facilities (Registers, Memories, Terminals, Clocks, etc.) and their parameters implied by the DDL description. The logic implied by the DDL description is completely described by the BEs and RTEs generated by the translator. The BEs generated by the translator are in the Sum of Products (SOP) form. The BEs in the DDL description that were not in the SOP form are throughout unchanged by the translator. Data Structures and the Translation Algorithm are described in this section.

3.2.1 Data Structures [15]

The characters of a description are read in the 80 column punched

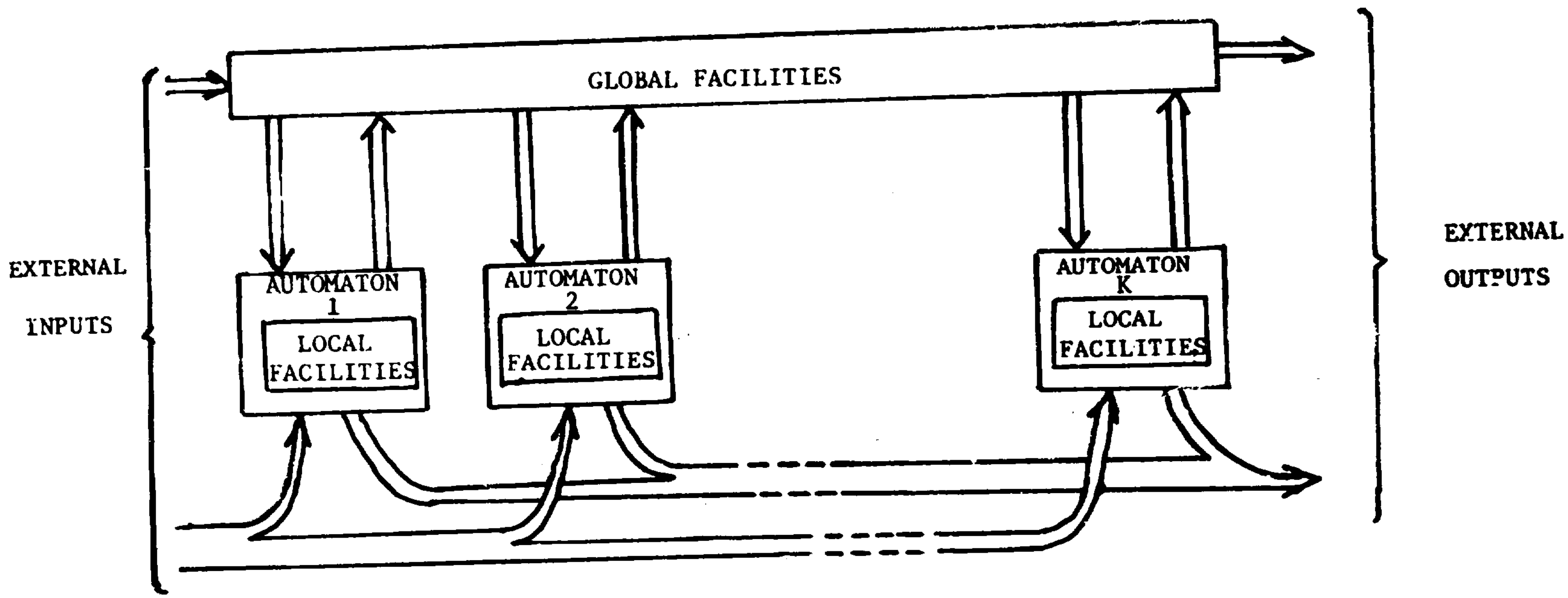


Figure 5: Digital System Model

card format. As these cards are read, blanks are ignored, and operator symbols are encoded as negative integers. The information provided by declaration statements (e.g. RE, TE, etc.) are accumulated in a facility table. The statements with connection and transfer operators and enclosing DDL syntax are placed in a single linked list. The main data structure consists of a facility table and linked list.

The current dimension of the facility table is 300 [10]. A combination of primary and secondary facilities can not be greater than 200 in a description. The columns of the F (Facility) table hold varying information as translation proceeds. The contents of the table are outlined below [15] (Note: F(I,J)-- I^{th} row, J^{th} column of the F table):

F(I,1),F(I,2)--Facility name in 2A4 format.

F(I,3)--Left subscript of RE, TE, DE, etc.
Number of words of memory.
For state entries, pointer to the binary string facility that expresses state assignment.
Third word in A4 format in binary string facilities.

F(I,4)--Right subscript for registers, terminals, etc.
Number of bits per word of memories.
Magnitude of binary strings.

F(I,5)--Total dimension of registers, terminals, etc.. The sign determines whether subscript ascends to the left or right.
Number of bits per word of memories.
Length of binary string.

F(I,6)--Negative entries indicate facility type.
Positive entries are pointers to primary facilities.

F(I,7)--Pointer to the innermost enclosing facility.

F(I,8)--Starting point of facility operation statement.

F(I,9)--End point of condition on operation statement.

F(I,10)-End point of operation statement.

Linked List (LL) of DDLTRN program is an array of dimension 2000. LL stores operation statements of a description. Unused LL cells are

linked in an available pool. Cells are taken from the top of the available pool to expand the main list. The deleted cells from the main list are linked back at the end of the available pool. Cells of the main list may contain [15]:

(1) A character in Character format.

(2) A subscript (integer biased by 201)

(3) A pointer to the F table (integer in the range 1 to 200), or

(4) An encoded operator or punctuation (negative integer).

Several minor data structures are also used in DDLTRN. Input-output routines make use of a 135 word array. A push down stack determines the nesting of declarations. Name entries in the F table are located using 129 word hash array during the early phase of processing. Later this array is used as a general work array [15].

3.2.2 Translation Algorithm [4,15]

DDLTRN program was developed by breaking the translation process into six major subroutines. Each subroutine is referred here as a "Phase." Each phase consists of several minor subroutines. A significant amount of computing time is not consumed for any phase to make a scan, major or minor, of the LL. In addition, a LL or a portion of it is printed by a number of subroutines. These subroutines may be requested to list LL after each or any phase with a suitable Flag declaration whose details are shown in Table 2.

Phase 1

Phase 1 begins with reading a DDL description in a card (80A1) format. During this phase:

(1) the DDL description is transferred from a card image into

Table 2 - Flag Integers [4]

Flag	Significance	Default
1	Print Source Card Images	Set
2	Print Declared Facilities and Operations	"
3	Print DDL string after Pass 2	Reset
4	" " " " " 3	"
5	" " " " " 4	"
6	" " " " " 5	"
7	" " " " " 6	Set
8	Print F Table after Last Pass	Reset
9	Print Encoded string after Last Pass	"
10	Execute through Pass 2 only	"
11	" " " 3 "	"
12	" " " 4 "	"
13	" " " 5 "	"
14	" " " 6 "	Set

the facility table and LL. All blanks are ignored.

(2) Punctuation and operator symbols are assigned a negative integer by table look-up according to Table 3.

(3) <TI>, <RE>, <ME>, <LA>, <TE>, <DE> declarations type entries are removed from the DDL description.

(4) Declared primary facilities are entered in the facility table.

(5) Secondary names are also tabulated, but their entries are linked together to the primary name via positive entries in the sixth column.

(6) Identical primary names defined in the nested or parallel blocks are made unique by appending a double quote (") and integer. Note that identical names in the same blocks are rejected.

(7) COmment declaration and in line comments are also deleted.

(8) For <SY>, <AU>, <OP>, the name of the facility is entered into the facility table. The facility is pushed onto the stack.

(9) <BO> type is pushed onto the stack.

Names are tabulated in this phase by packing letters and digits into 2A4 format until punctuation or operator symbol is encountered. The subscript is converted from ASCII character codes to binary, and the total dimension and direction of ascendancy is determined. Note that names in Declared Operations have not yet been examined. (e.g., In the declaration <OP>ADD(3)X, X has not been examined yet.)

Table 3 - Codes for Punctuation and Operators [4]

Character	Code	Semantics	Character	Code	Semantics
'	-20		$	-40	Extension
.	-21		[	-41	Concatenation
:	-22		^	-42	NOT
;	-23		'	-43	Selection
]	-24	If	*/	-44	AND reduction
			^*/	-45	NAND reduction
!	-26	If (value)	^+/	-46	NOR reduction
#	-27	value	^@/	-47	XNOR reduction
$	-28	<OP>argument	@/	-48	XOR reduction
(	-29		+/	-49	OR reduction
)	-30		*	-50	AND
<	-31		^*	-51	NAND
-	-32		^+	-52	NOR
>	-33		^@	-53	XNOR
"	-34	not used	@	-54	XOR
∞	-35	For	+	-55	OR
/	-36				
?	-37	not used	=	-60	Connection
↑	-38	not used	<-	-61	Transfer
_	-39	not used	->	-62	Go to

Phase 2

Syntax reduction begins in this phase. While making the pass over the system description in LL:

(1) All secondary names appearing on the right of an equal sign in declarations (such as TErminal, REgister, etc.) are replaced with their primary equivalents.

(2) All identifiers are replaced with the string they identify.

(3) All names and binary strings (constants) are encoded (replaced with pointers to their facility table entries).

(4) Subscript punctuation are deleted and subscript are replaced with identifiable biased integers.

(5) The syntax of <OP>, <BO>, and <ST> declaration are removed.

(6) BOolean equations are moved to the head part of their enclosing <SY> and <AU> declarations.

(7) STate statement syntax is replaced with IF - THEN syntax.

(8) Compound BOolean expressions serving as conditions on operations are replaced with terminals of unit dimension.

(9) Operator calls with arguments are transferred to the connection statement.

Note that names in operation statements are packed from A1 to 2A4 format and subscripts are converted to binary.

Phase 3

A single pass is made over the linked list which is now fully encoded and free of all declaration types except SY and AU. During this pass:

(1) IF - THEN and IF - VALUE conditions are combined and distributed over the members of the set so that each operation

appears as the body of a simple IF - THEN clause.

(2) GO TO operations are converted to conditional transfers of a constant (the state assignment) to the state sequencing register (the enclosing automaton).

(3) If the LL scan reveals on <AU> code, the name of the automaton is examined. If the global condition is <TI> facility, that facility is used as the clock variable. If the global condition is a Boolean expression, that expression is formed into a Boolean equation and created terminal is used as clock. (e.g. <AU><AU1:A(1)*A(2):, terminal "1 is created and used as clock where, "1=A(1)*A(2).) Clocking condition is distributed on all register transfer and memory operations within the Automaton declaration.

At the end of phase 3 system description consists of BEs and conditioned register transfer operations.

Phase 4

Again LL is scanned. The starting point of an operation statement is retained as the statement is scanned in a search for concatenation operators. If no concatenation operator is found, the next operation statement is examined. If the statement includes a concatenation, the scan pointer is set back to the beginning of the statement. Then a second scan is made. All concatenation operators are eliminated by breaking operations into operations on subfacilities formed by partitioning operand facilities according to the dimensions of the concatenation operands. Concatenated operands of reduction operators are not modified by the DDL translator.

Phase 5

It is very useful for two or more connection or transfer statements with identical information sink operands to appear in a description. To obtain the total logic of that sink facility, such portion of the descriptions must be gathered into one connection transfer statement. During phase 5 all connection and transfer operations with the same data sink (left operand) are gathered into one compound operation.

Phase 6

Facilities with subfacilities serving as data sinks of connection and transfer operations are broken into disjoint subfacilities and a right hand side of a connection or transfer operation is formed for each new subfacility. Phase 6 provides the Boolean equations and Register transfer statements from which the system may be synthesized.

Example outputs of each of these phases is provided in Chapter VI and also in [4].

3.3 THE SIMULATOR (DDLSIM) [4,16]

DDLSIM is a program for simulating the digital system described in DDL. The simulator has a simple, powerful and completely free-format command language. This language provides the user with complete control over the simulation process. DDL system description modification(s) are not required during the simulation process. DDLSIM does a through error checking of described systems, simulation control statements and the simulation process itself. Self explanatory messages that pin-point errors are issued.

Digital systems to be simulated are first described in DDL. This description is translated by DDLTRN. The BE, RTE and the facility table generated by DDLTRN provide the system description required by the

DDLSIM. The description is pre-processed by the simulator to establish data structures and tables that permit more efficient and controlled simulation. The information for controlling simulation is provided by using simulator command language. The DDLSIM command language consists of twelve different types of declarations for supplying parameters, input data, options and other control information necessary for simulation. The language is largely free of format restrictions. Card images are scanned in turn from left to right. Any declarations may start at any point and end at other points in the card deck. A declaration can be continued on as many cards as necessary. More than one declaration can be supplied on the same card. At the start of a new declaration, the previous declaration is ended automatically. An 'End Of File' statement terminates the simulation. In general, the order of the declaration is not important. More than one declaration of the same type can be used.

Statements of the simulation deck are sequentially processed by the simulator. Each simulator declaration has the general syntax [16]:

<Declaration-id> Body.

Only the first two characters of the Declaration-id are examined by the simulator. All declarations, except the SImulate have a Body following the heading. The syntax diagrams of the twelve different types of declarations are given in Appendix B. For the details of the simulator command language refer to [4,16].

3.4 SUMMARY

The output of DDLTRN can be used for the hardware implementation of the system. The HSA is described in Chapter 4. Designer simulates design and refines it at a high level using DDL description till all design

errors are corrected. The design now is ready for implementation.

CHAPTER 4. THE LOGIC SYNTHESIS ALGORITHM

The output of DDLTRN consists of BEs corresponding to the combinational logic and RTEs corresponding to the sequential logic of the digital system described in DDL. An algorithm designed to synthesize this system using the CADAT system standard cells is described in this chapter. The algorithm requires that the BEs be in the SOP form. Hence, the BE output of DDLTRN is converted into the SOP form. The RTEs are synthesized in two parts: one corresponding to the CONDITION and the other corresponding to the TRANSFER. This chapter describes the combinational synthesis algorithm first. Next sequential logic synthesis is discussed, followed by the overall synthesis algorithm.

4.1 COMBINATIONAL LOGIC SYNTHESIS ALGORITHM [1]

The combinational logic synthesis algorithm is described below. Let the number of digits in the literal/product pattern (LPP) be n, and K_i is the i^{th} digit of the LPP.

(1) Scan the Boolean function to be implemented and count the number of literals in each product term to derive the digits of the LPP. Reduce each product term with more than 2 literals into a term with 1 literal (i.e. for $K_i = 3$ or 4, implement using AND cell with proper number of inputs; If $K_i > 4$, implement using more than one AND gate.). Generate LPP for the function from the above derived (and reduced) digits.

(2) Arrange the LPP in descending order of its component

digits. (e.g. 2121 is arranged as 2211).

(3) For $n > 4$, LPP is split into several 4-digit units (last unit can have less than 4 digits). Each unit is implemented independently, thus reducing the unit to a 1 in the original LPP. To implement the unit with n digits ($n \leq 4$), it is compared with all the n digit standard cell LPPs. The unit is implemented using the standard cell whose LPP has minimum number of mismatches according to the following rules:

(a) If the unit is that of a pure SUM term (i.e., $k_i = 1$ for all $i = 1$ to n, $n \leq 4$); implement using OR cell with proper number of inputs. Implement above unit as NOR cell instead of OR cell for the LPP in which $K_i = 2$ for any $i = 1$ to n, $n > 4$. (For example, in the LPP = 22221111 unit 1111 is implemented using 4 input NOR gate).

(b) If in the unit, $K_i = 2$ for any $i = 1$ to n, $n \leq 4$, the mismatched digit is incremented by 1 to create the standard cell LPP.

Example: 2221 is implemented as 2222.
2111 is implemented as 2211.
221 is implemented as 222.
211 is implemented as 222.
21 is implemented as 22.

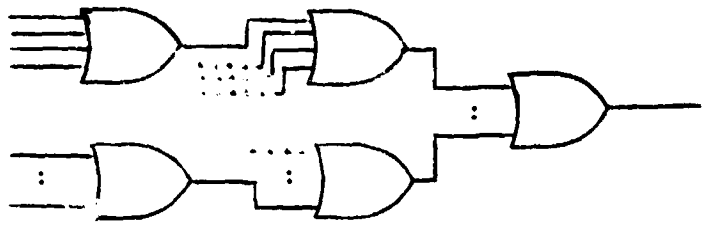
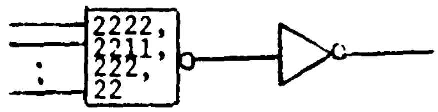
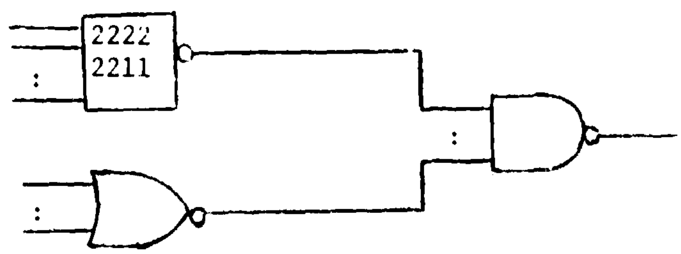
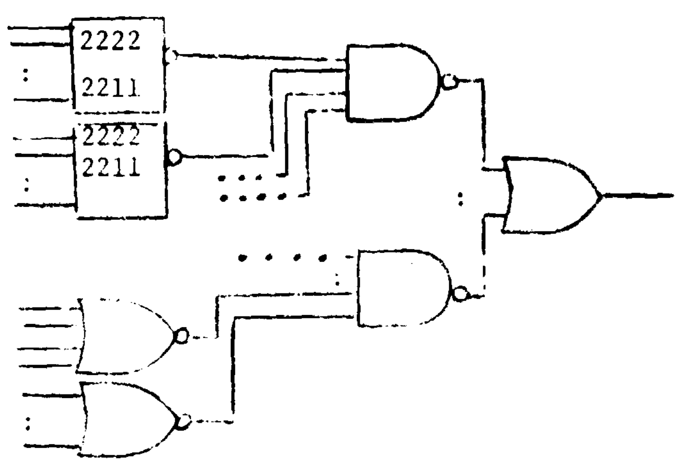
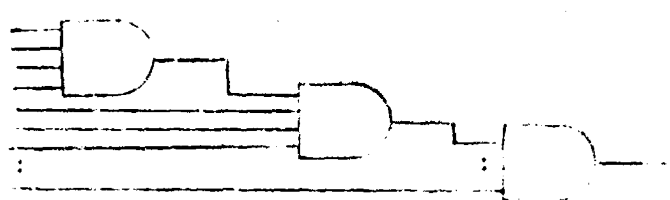
(4) Final implementation requires different actions for the different LPPs, as summarized in Table 4.

(a) The output must be inverted for the LPP in which $K_i = 2$ for any $i = 1$ to n, $n \leq 4$.

Table 4 - Final Implementation

n - Number of digits in LPP

$K_i = i^{th}$ digit of LPP, $i = 1$ to n.

LPP configuration	Implementation
$K_i=1$, for all $i=1$ to n, $n \le 4$.	
$K_i=1$, for all $i=1$ to n, $n>4$.	
$K_i=2$, for any $i=1$ to n, $n \le 4$.	
$K_i=2$, for any $i=1$ to n, $4<n\le 16$	
$K_i=2$, for any $i=1$ to n, $n>16$.	
$k_i \ge 2$, for $n=1$	

(b) The synthesis is complete for the LPP in which $K_i = 1$ for all $i = 1$ to n, $n \leq 4$.

(c) The stored outputs during step 3 (stored after implementation of every 4-digit unit in which $k_i = 1$ for all $i = 1$ to n, $n > 4$) are implemented using OR cells with the proper number of inputs.

(d) The stored outputs during step 3 (stored after implementation of every 4-digit unit in which $K_i = 2$ for any $i = 1$ to n, $n > 4$) are implemented using NAND cells with the proper number of inputs. For $n > 16$, only one NAND cell is required, as shown in Table 4). Output of NAND cells are implemented using OR cells with proper number of inputs.

(e) If LPP is that of a PRODUCT term ($K_i \geq 2$, $n = 1$), implement using AND gates (single or multiple stages).

Note that the above algorithm does not minimize the Boolean function. Only the literals are counted, not the actual number of input variables needed.

Implementation Examples:

X=A+B*D+M*N*P+Q*R*S,

Step 1	1	2	3	3
	1	2	1	1
Step 2	2	1	1	1
Step 3-b	2	2	1	1
Step 4-a	Output of 2211 must be inverted.			

The above example can be implemented with 1360, two 1630's, and one 1310 cells. The cost is (30 devices, 33 mils) and can be implemented as shown in Figure 6.

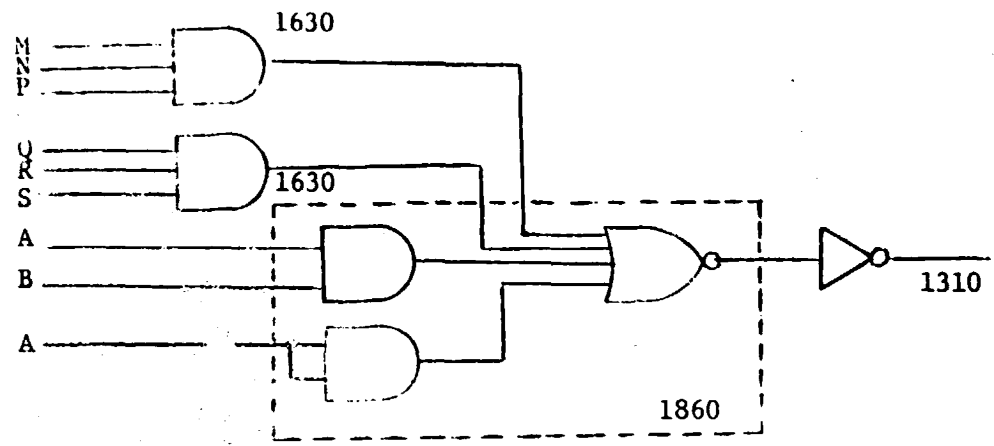


Figure 6: Implementation for X=A+B*D+M*N*P+Q*R*S

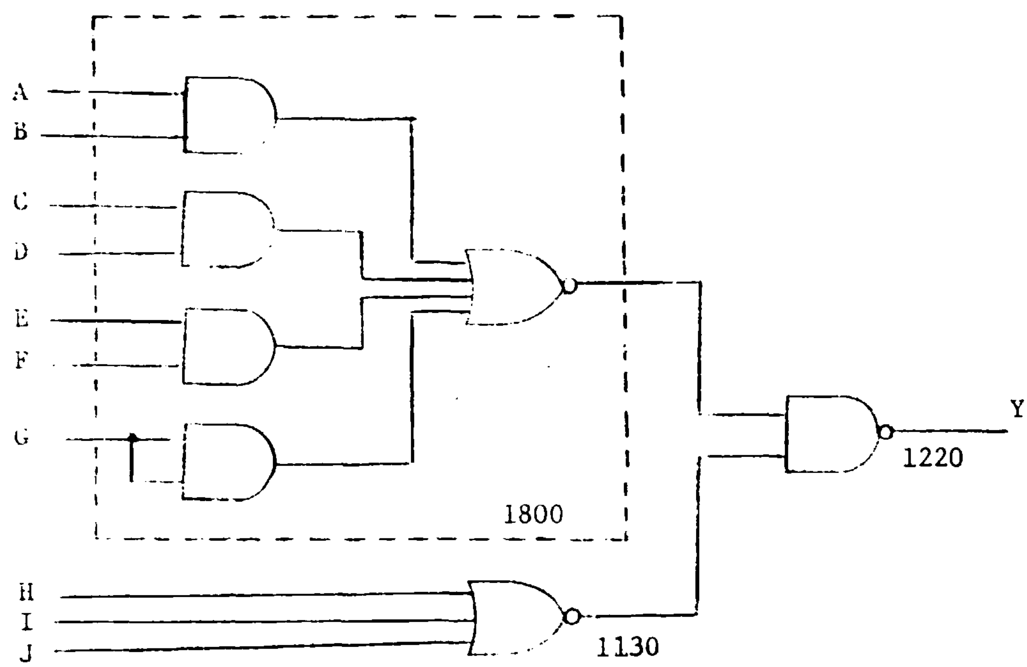


Figure 7: Implementation for Y=A*B+C*D+E*F+G+H+I+J

```
              Y=A*B+C*D+E*F+G+H+I+J,
Step 1          2   2   2  1 1 1 1
Step 2          2   2   2  1 1 1 1 (no action necessary)
Step 3          2   2   2  2 1 1 1
Step 4-d        Output of 2222 and 111 are input to 2-input NAND.
```

The above example can be implemented with one 1800, one 1130, and one 1220 cells. The cost is (26 devices, 30.7 mils) and can be implemented as shown in Figure 7.

4.1.1 Equation involving Reduction and Selection

Terms involving AND-reduction and selection are identified as "decoders" and are replaced with appropriate logic products. The following steps are performed to reduce them to SOP form, which can be synthesized using the above combinational logic synthesis algorithm:

(1) Synthesize the selection operator by complementing the bits of its left operand if a zero appears in the correponding position of the right operand.

(2) Provide the operator defined in the reduction operator between each bit of the selected left operand.

```
Example:        B=*/A'OD2,   (A is two bits wide).
                B-*/A'00,
         Step-1 B=*/^A(1)^A(2),
         Step-2 B=^A(1)*^A(2),
```

Terms involving only Reduction Operators are also reduced to the SOP form by the above process, except that the selection is not involved.

4.1.2 Constants

Constants do not denote physical entities, but they do affect the treatment of the equations in which they appear. During synthesis constants drop out of the design. Table 5 shows the action taken to remove the constant from the BEs [18].

Table 5 - Logic Simplification [18]

Constant Encoded	Operator	Action taken on constant	Action taken on operator	Action taken on operand
0	*	Removed	Removed	Removed
1	*	Removed	Removed	None
0	+	Removed	Removed	None
1	+	None	Removed	Removed
0	@	Removed	Removed	None
1	@	Removed	Removed	Inverted

Examples: A=B*C+D*1D1 correspond to A=B*C+D.
A=B*C+D*0D1 correspond to A=B*C.
A=B*C+D@0D1 correspond to A=B*C+D.
A=B*C+D@1D1 correspond to A=B*C+^D.
A=B*C+0D1 correspond to A=B*C.
A=B*C+1D1 correspond to A=1.

A, B, C, D in the above examples are each 1 bit long.

Example: A=B*C+D*1D2 (A, B, C, D are each two bits)
correspond to the following two equations:
A(1)=B(1)*C(1)+D(1),
A(2)=B(2)*C(2).

4.1.3 EXclusive-OR (EX-OR)

An EX-OR cell is available in the standard cell library. EX-OR is synthesized using this cell as shown in Figure 8 for the examples X=A@B and Y=A@B@C@D.

4.1.4 Equations with Parenthesis

If a portion of the BE is in the parenthesis, that portion is synthesized first as in Combinational Synthesis Algorithm. The rest of the equation is synthesized next. For nested parenthesis, innermost parenthesis is synthesized first.

4.2 SEQUENTIAL LOGIC SYNTHESIS ALGORITHM

DDL generates RTE corresponding to sequential logic. Each RTE is of the form:]CONDITION] TRANSFER.

The "CONDITION" and "TRANSFER" portions in turn are BEs. Each of these portions are separately synthesized by using the above algorithm. The "CONDITION" portion provides the clock input; the right hand side of the "TRANSFER" provides the source and the left hand side of the "TRANSFER" provides the destination register. A clocked D flip-flop is assumed for each bit of all the registers.

Example:]A*P+B*P]K<-A*C+B*D, implies the logic circuit shown in Figure 9.

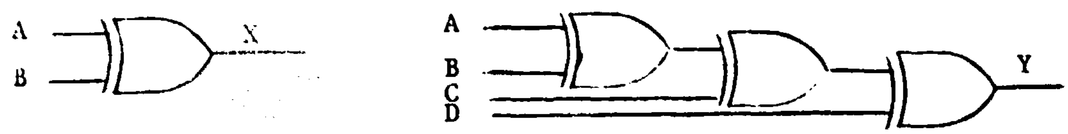


Figure 8: EX-OR Implementations

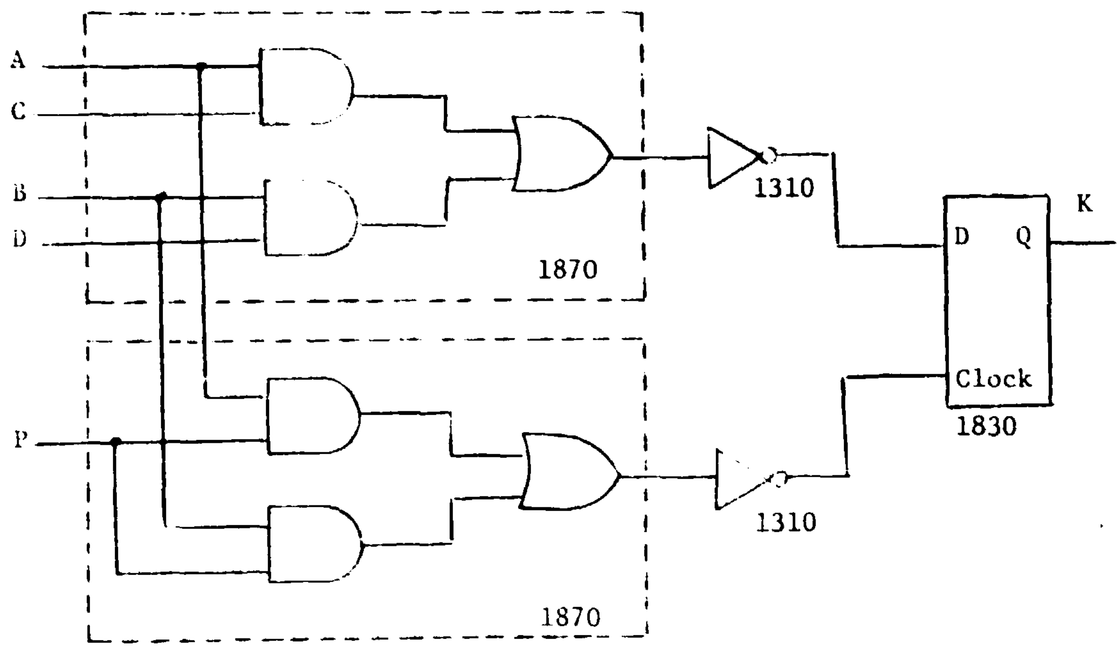


Figure 9: RTE Implementation

4.2.1 Memory

Memory references in the DDL description is of the form M(MAR), where MAR is the same register (assumed to contain the address) in all references to the memory M. The memory references will generate either a Memory Read or a Memory Write signal. The memory reference on the right hand side of the RTE designates a Memory Read, while the memory reference on the left side of the equation designates a Memory Write. The memory model assumed for the synthesis is shown in Figure 10.

Example:]A*P+B*P]M(MAR)<-A*MBUS+B*MBUS. MAR is memory address register. MBUS is the terminal through which Memory Read and Memory Write are performed. The dimension of MAR is a function of the number of words in the memory, and MBUS must have the same number of bits as the memory word. A and B are single bit facilities. The hardware implementation is shown in Figure 11.

4.3 OVERALL SYNTHESIS ALGORITHM

The synthesis algorithm follows the following sequence of actions:

(1) Memory references from Memory Equations are reduced to Memory READ and WRITE signals.

(2) RTEs are broken into two BEs corresponding to "CONDITION" and "TRANSFER" portions.

(3) Equations with Selection and Reduction operators are reduced to SOP form.

(4) EX-OR operators, constants and parenthesis from the equations are eliminated.

(5) BEs in SOP form are synthesized using the Combinational Logic Synthesis Algorithm.

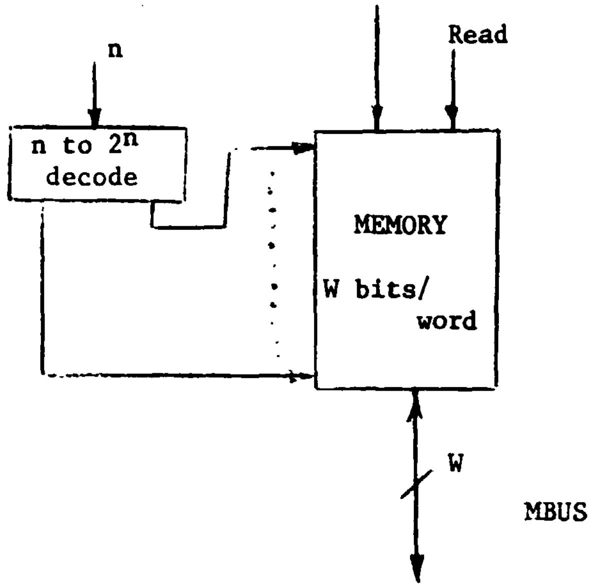


Figure 10: Memory Module

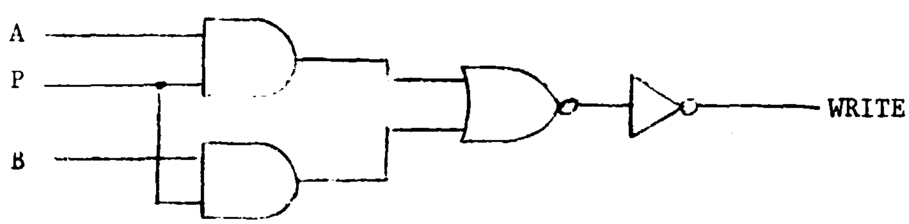


Figure 11: Memory Write Implementation

4.4 SUMMARY

An algorithm for selecting standard cells for implementing combinational and sequential logic is presented. The algorithm is suitable for implementation as a computer program. The algorithm is general enough to be useful in any LSI design environment. Logic minimization aspects are not considered, although simplifications are done when constants are involved. The equations from DDLTRN output are implemented one at a time. Similar logic synthesis algorithms for other HDLs are available [19,20].

CHAPTER 5. IMPLEMENTATION DETAILS

The HSA described in Chapter 4 is implemented as a FORTRAN program on SEL-32 computer system of NASA-MSFC Electronics and Controls Laboratory. This chapter provides the implementation details. The data structures used in the program are discussed followed by the details of each of the support programs (subroutines). The program flowcharts are included in Appendix C.

5.1 DATA STRUCTURES

A new subroutine (DDSYN) added to DDLTRN formulates 3-disk resident data files to provide the input required by DDLSYN. The F table is transferred as it is, as the first data file (DATA1). The other two data files (DATA2 and DATA3) are created from LL of DDLTRN. LL is split into DATA2 and DATA3 to separate the mixed formats for subscripts, constants, pointers to facility table and encoded operator symbols (negative integer). Also, DATA2 and DATA3 arrange the BEs and RTEs in a sequence, so that no LL pointer information is required to access these equations. DATA2 file contains the following information for each equation:

(1) Identifier and operator symbols in the character (A4) format,

(2) Subscripts as blanks, and

(3) Constants in the character (A4) format.

DATA3 file contains the following information for each equation:

(1) Pointer to facility table entry for each identifier and

constant,

(2) Negative integer for operator symbols as defined in Table 3,

and

(3) Subscripts biased by 201.

The use of arrays and variables used in the program are described in Table 6 and Table 7 respectively. The arrays QU2 and QU3 constitute the Identifier Table. QU2 contains identifier name in the character (2A4) format and QU3 contains subscript for that identifier in the integer format. Each identifier appearing in the equation is defined in the identifier table. The identifier is replaced in the equation by a pointer pointing to the identifier table entry as shown in Figure 12. There will be one net for each identifier defined in the identifier table. Net contains connectivity information for that identifier (see Figure 13). The generated net file of which each record corresponds to a net (which later can be used as input to the Placement and Routing 2-Dimensional (PR2D) program of the CADAT system) is maintained throughout the synthesis process on the disk to minimize the core usage. The Net is accessed randomly by using the identifier pointer as the record number. The record format for each identifier is shown in Table 8. The first word of the record is the identifier pointer pointing to the corresponding net in the identifier table. The second word of the record gives the number of entries already made. Each entry is made up of cell number and pin number information. Only 10 entries are allowed per record. A driving buffer is provided in the net if it has more than 9 entries. The tenth entry is defined as the input to the driving buffer. Output of the driving buffer is defined as a new identifier in the

TABLE 6 - LIST OF ARRAYS

FACL	-	Facility table as described in 3.2.1.
IRE	-	Current equation being synthesized in character format.
IDU2	-	Current equation being synthesized: (a) A pointer to the facility table for the identifier (b) An encoded operator or punctuation (negative integer) (c) A subscript (integer biased by 201).
ILRL	-	Contains each digit of the LPP for the equation being synthesized.
STK	-	Stack contains operands in the equation.
NEXT	-	Inputs to the selected standard cell are stored here.
QU2	-	This array has identifier in the character format.
QU3	-	This array has subscript information for the identifier in QU2 (integer format).
GATPE	-	Selected standard cell during synthesis are stored here.
SYOU	-	The output identifier of selected standard cells after every 4 digits for the unit's of SOP expression.
MSTR		Used for temporary storing equation when equation in parenthesis being synthesized.
ICH	-	Current equation being synthesized: (a) A pointer to the identifier in integer format (b) Operator symbols in character format (c) Constants as negative integer.
PCONE	-	Common inputs found during synthesis are stored here.
SCEL	-	Selected standard cells for the first equation of the multibit equation are stored here.
JLRL1	-	Number of digits in the LPP of the standard cell.
JLRL2	-	LPP (as in column 6 of Table 1) for the standard cell in JLRL1.
JLRL3	-	Cell number (as in column 1 of Table 1) for the standard cell in JLRL1.
JMATCH	-	Number of matches for the LPP with the standard cell LPP and standard cell number are stored here.
MATCH	-	Each digit of the LPP for the unit for which standard cell is to be selected are stored here.

TABLE 7 - LIST OF VARIABLES

TOP	-	Stack pointer.
IZ	-	Number of digits in LPP being synthesized.
FL	-	Set when equation in parenthesis being synthesized.
NLP	-	Number of left parenthesis in the equation.
NRP	-	Number of right parenthesis in the equation.
COUNT	-	Number of product terms in the digit of LPP.
LRAL	-	LPP for the equation.
ID	-	Number of records allocated in Net table.
IC	-	Size of the equation being synthesized.
NEXOR	-	Number of EX-OR symbols in the equation.
ICEL	-	Number of cells stored in SCEL array.
CELL	-	Number of cells selected for the design.
NCEL	-	Number of cells selected for the current equation.
POC	-	Number of entries in the PCONE.
IPNTR	-	Pointer pointing to the equation which will be synthesized next.
NEQ	-	Reset when no more equation to be synthesized.
ADD	-	Number of equations in the multibit equation.
INPIN	-	Input pin number of standard cell.
OUPIN	-	Output pin number of standard cell.
ILZ	-	Number of digits in standard cell LPP required to select standard cell.
FBIT	-	First bit of the multibit equation.
LBIT	-	Last bit of the multibit equation.
NBIT	-	Number of bits of the equation.
ANDEC	-	Number of AND reduction symbols in the equation.
ORDEC	-	Number of OR reduction symbols in the equation.
SELC	-	Number of selection symbols in the equation.
FL2	-	Set, if any digit of LPP is 2.
FLAB	-	Set, if LPP is greater than 4.
FLPE	-	Set, if equation in parenthesis needed to be inverted.
FRONT	-	Number of entries in SYOU array.
RTE	-	RTE=2, when condition part of RTE equation is synthesized. RTE=1, when transfer part of RTE equation being synthesized. RTE=0, when equation being synthesized is not RTE.
FLEX	-	EX-OR is selected when set.
IRFF	-	D-flip-flop is selected if set.
CLK	-	Clock input to the flip-flop.
MEMO	-	Set, when memory read equation being synthesized.
NDIG	-	Maximum number of digits in the standard cell LPP. (NDIG is 4 for current version.)
NBUFR	-	Driving buffer is to be provided after number of entries, (NBUFR is 10 for the current version.)

Identifier Table

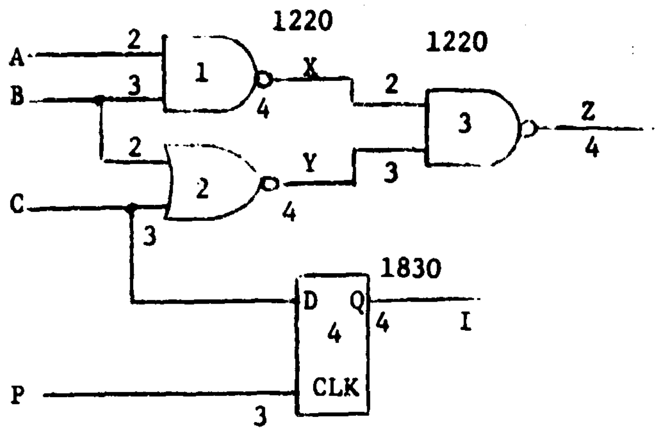
Identifier Pointer	Identifier
1	X(1)
2	A(1)
3	B(1)

X = A*B
1 = 2*3
↑ ↑ ↑
Pointer

Figure 12: Identifier Table Representation

Table 8: Record Format

Word	Entry	
1		Identifier Pointer
2		Number of Entries in Record
3	Entry	Cell Number
4	1	Pin Number
.	.	.
.	.	.
.	.	.
21	Entry	Cell Number
22	10	Pin Number



IDENTIFIER TABLE

Identifier Pointer	Identifier
1	A(1)
2	B(1)
3	X(1)
4	C(1)
5	Y(1)
6	Z(1)
7	P(1)
8	I(1)

CELL TABLE

Cell No.	Standard Cell No.
1	1220
2	1120
3	1220
4	1830

NET TABLE

Same as Identifier pointer

Net	Cell	Pin	Cell	Pin
1	1	2		
2	1	3	2	2
3	2	4		
4	2	3	4	2
5	3	3		
6	3	4		
7	4	3		
8	4	4		

Figure 13: Synthesis Output

identifier table. The first entry in the net of the new identifier is defined as the output of the driving buffer. In the original net (record) the first word is replaced by a pointer pointing to the newly defined identifier in the identifier table. If the first word of a record does not match the record number, it is a pointer to the record in which the net list continues, as in Figure 14 (new entries for Net A are made in the record number 15).

The cell table contains the standard cell number (column 1 of Table 1) from the library. In the net, cell number is defined by the pointer pointing to the cell table (GATPE array) where standard cell number is stored. Figure 13 shows implementation and corresponding entries in the cell table for selected cells.

5.2 SUPPORT PROGRAMS

Each equation in the DDLTRN output during phase 6 is synthesized separately. The structure of the program is shown in Figure 15. In this section, details of each subprogram are given.

<u>Subroutine IREAD</u>

This subroutine provides input to the synthesis program. One equation from DATA2 is read into IRE array. Corresponding information for the equation is read into IDU2 array from DATA3. This subroutine is called again at the end of the synthesis of an equation to read a new equation. Flag NEQ is reset when no more equations are left to be synthesized for the system.

<u>Subroutine SELECT</u>

This subroutine implements decoder type of equations in the IRE array. (Many selection types of equations make a decoder. Each selection type of equation corresponds to one output of the decoder). The

Record Number 2
(before driving buffer)

First word	2
	9
Entry	1
1	2
Entry	2
2	3
.	.
.	.
.	.
Entry	16
9	4
Entry	0
10	0

Identifier Table

Identifier No.	Identifier
1	X(1)
2	A(1)
3	XXXXXXXX(1)
.	
.	
.	
14	Y(1)

Record Number 2
(after driving buffer)

First word	15
	10
Entry	1
1	2
Entry	2
2	3
.	.
.	.
.	.
Entry	16
9	4
Entry	18
10	2

Identifier Table

Identifier No.	Identifier
1	X(1)
2	A(1)
3	XXXXXXXX(1)
.	.
.	.
.	.
14	Y(1)
15	XXXXXXXX(1)

Figure 14: Driving Buffer Representation

```
MAIN --
        IREAD
        MEMW
        MEMW
        GENIRE
        SELECT --
                      SELEC1
                      SEARCH
                      SIGN
        REDUCT --
                      SELEC1
                      SEARCH
                      SIGN
        CRICH
        SYNIN1 --
                      SYNIN --
                                   POP
                                   SCL
                                   NET
                                   NETOU
                                   PEXOR --
                                                SCL
                                                NET
                                                NETOU
                                   ATNG
                                   SYOU
                                   SCL
                                   NET
                                   NETOU
                                   CONEC --
                                                SCL
                                                NET
        PRCONE                                  NETOU
        OUNET
```

Figure 15: Program Structure

constant with selection operator is in the character format. It must be first converted to the integer format. Next, inverters are provided for those bit positions of selection operator's left operand in which 0's are found in the right operand (constant) [19,21]. The operator symbols defined by the reduction operator is provided for each bit position of the operand by calling SIGN subroutine. At the end of the subroutine, the ICH array contains the equation in SOP form for the selection type of equation in IRE array.

Subroutine SELEC1

This subroutine expands the multibit identifier in reduction and selection type equations in IRE array into ICH array. Each bit of the multibit identifier is defined in the identifier table. If subscript range is not defined with the multibit identifier, this subroutine uses the corresponding facility table entry to define the subscript range. ICH array will have multibit identifier expanded for each bit in a pointer list form by the pointers where each bit is designated by a pointer to the bit in the identifier table.

Subroutine REDUCT

This subroutine implements reduction type of equations in the IRE array. It calls SELEC1 for multibit operands. Then operator symbol defined in the reduction operator is provided for each bit position of the operand by calling the SIGN subroutine. At the end of this subroutine, ICH array will contain the equation in SOP form.

Subroutine SEARCH

This subroutine is called everytime an identifier is found during the synthesis. It enters the identifier into the identifier table, only if it has not been entered previously. Otherwise, it returns the

integer pointer pointing to the corresponding entry in the identifier table, to the calling routine.

Subroutine SIGN

This subroutine provides AND, OR and EX-OR operator symbols between two operands defined by the reduction operator (*/,+/,@/). For example,

*/A = A(1) A(2) A(3) (equation after SELEC1)

*/A = A(1)*A(2)*A(3) (equation after SIGN)

where A is defined to be 3 bits long.

Subroutine GENIRE

This subroutine provides subscript range in IDU2 array (subscripts biosed by 201), and corresponding blanks in the IRE array, for the multibit equations defined without subscript range. The subscript range for multibit identifier is found from the facility table. During synthesis for the blanks in the IRE array, the subscript range is obtained from the IDU2 array.

Subroutine POP (KOUT)

This subroutine pops the stack KOUT number of times.

Subroutine SCL

This subroutine finds the best match for LPP by counting the number of matches for each standard cell LPP. The best matched standard cell is selected for implementation. ILZ gives the number of digits in the LPP. Only the ILZ digit standard cell LPPs are compared. The above procedure is not used to select D-flip-flop or EX-OR. The calling subroutine sets IRFF to select D Flip-flop, and FLEX to select EX-OR. The D-flip-flop or EX-OR is selected by the subroutine SCL if IRFF or FLEX is set respectively.

Subroutine PEXOR

This subroutine implements EX-OR gate with two operands. If complement symbol precedes to the left or right operand of EX-OR equation, first inverter and next EX-OR gate is implemented. For example, in the equation ^A$\oplus$B, A$\oplus$^B, inverters for A and B are implemented first. If one operand is a constant, then EX-OR gate is not implemented. Another operand is complemented or not complemented for constant equal to "1" or "0" respectively [21]. For example, A@0 = A and A@1 - ^A.

Subroutine MEMR

This subroutine eliminates memory references on the right of the connection or transfer operator. ICH array will have memory read equation at the end of the subroutine.

Subroutine MEMW

This subroutine eliminates all multibit references in the memory write equation to the right of the connection or transfer operator. IRE array has this new equation without any multibit memory references.

Subroutine CRICH

This subroutine derives ICH array from the array IRE and IDU2. Each identifier is defined by integer pointer pointing to the identifier table. The constants are identified as negative integers to differentiate between constants and integer pointers. Operator symbols are passed as they are. During this subroutine, the number of left and right parenthesis and the number of EX-ORs are counted in the equation being synthesized. For a multibit equation the number of times the equation needs to be synthesized is also calculated from the subscript range.

Subroutine SYNIN

This subroutine scans the equation (BE) to be implemented and counts the number of literals in each product term for the function. For example, equation Z = A+B*D+M*N+P*Q, LPP is defined as 1222. It pushes the identifier pointers corresponding to the identifiers in the product term corresponding to each digit of the LPP into the stack as it generates the LPP. Any digit of LPP greater than 2 is reduced to less than or equal to 2 by providing an AND gate. For example,

Z = A+B*D*M*N*P*Q is reduced to

Z = A + OU2 as shown in Figure 16.

If parenthesis exist in the equation, then the LPP for the equation in the innermost parenthesis is generated first. For example in: Z = A+^(B*D+M*N). The LPP 22 corresponding to the equation within the parenthesis is generated first. Constants with AND or OR operators are eliminated in this subroutine as described in section 4.1.2.

Subroutine SYNIN 1

This subroutine derives LPP for the Boolean function in ICH array by calling SYNIN. If LPP has greater than NDIG digits, then LPP is split into several NDIG units (NDIG = 4 in the current version). Each unit is implemented independently by calling SCL subroutine, thus reducing the unit to the single digit 1 in the original LPP. The output of each unit is stored in SYOU array for final implementation. For the Boolean function with the parenthesis, LPP for the function in the innermost parenthesis is derived by calling SYNIN, and then implemented. The equation in the parenthesis is replaced by the required standard cells. This process is repeated till no parenthesis are left in an equation. For example, Z = A+^(B*D+M*N). Then first LPP 22 must be

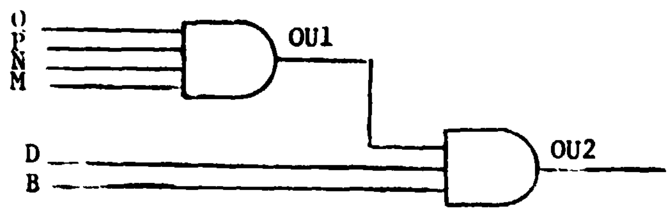


Figure 16: Implementation of B*D*M*N*P*Q

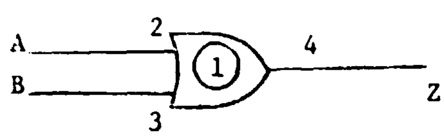


Figure 17: Implementation of Z=A+B

implemented using standard cell 1870. Next ^(B*D+M*N) is replaced by output (OU1) of the 1870. Now the equation is reduced to Z = A + OU1. Next LPP il corresponding to this equation is implemented using the standard cell 1720 (OR gate). Standard cell is selected by calling SCL subroutine. For the multibit equation, first the equation is implemented as above. All the cells selected during implementation of the first bit are stored in the SCEL array. The synthesis of the subsequent bits in the multibit equation uses these stored standard cells in SCEL array without repeating the SCL subroutine.

Subroutine SYOUT

This subroutine expands the function LPP to match the number of inputs needed for the selected standard cells for the LPP. For example, Z = B*D+M*N+P*Q+A has the LPP 2221. The best match for this function is standard cell LPP 2222 (standard cell number 1800) as found in SCL subroutine. Each digit of the LPP and the standard cell LPP are compared. The digits of the LPP defines the number of times stack is to be popped out. If the digit of the LPP and standard cell LPP are not the same, the popped out identifier pointer is duplicated as input to the standard cell. The last digit of LPP 2221 does not match with the standard cell LPP 2222. The product term (identifier pointer for A) corresponding to the last digit of LPP must be duplicated in the original function. The input to the standard cell must be: A,A,Q,P,N,M,D,B.

Subroutine ARNG

This subroutine arranges LPP defined by SYNIN subroutine into an ascenting order. It also rearranges the stack according to the ordered LPP.

Subroutine NET

This subroutine formulates the Net for each input to the standard cell by storing input pin number and cell number in each entry of the record for that Net. Cell number is the pointer pointing to the standard cell in the cell table. The Record corresponding to the identifier is accessed randomly from the disk by the record number as discussed in Section 5.1. The Record number is the pointer pointing to the identifier in the identifier table. For example, Z = A+B requires 1720 (OR gate) as standard cell, as shown in Figure 17. In the Net of identifier A, cell number 1 and input pin number 2 are stored as one entry. In the Net of identifier B, cell number 1 and input pin number 3 are stored as one entry. This subroutine also provides a driving buffer if there are more than 10 entries for the Net as described in Section 5.1.

Subroutine NETOU

This subroutine formulates the Net for the output signal of the selected standard cell by storing the output pin number and cell number. The Record corresponding to the identifier is accessed randomly from the disk by the record number. For the above example, in the net of output identifier Z, cell number 1 and outpin number 4 are stored. This subroutine also provides the driving buffers for the Net as described in Section 5.1.

Subroutine PRCONE

If the equation reduces to X =A, identifier X is the same as A. So A in all the other equations for the system must be replaced by X. In the synthesis process, this is not done till the complete synthesis is over, but X and A are stored in the array PCONE. At the end of the synthesis, the Nets for X and A are accessed randomly. All entries of

A are initialized to zero. This process is repeated for all the entries of PCONE.

Subroutine OUNET

Net table is disk-resident. To print out the Net table, each record of the Net table is accessed randomly. Each record has 10 entries and an entry is made up of a cell number and a pin number (as discussed in Section 5.1). The subroutine OUNET prints out the Net table at the end of the synthesis.

Subroutine CONEC

This subroutine provides the final implementation for LPP as shown in Table 9. FL2 is set when $K_i = 2$ for any $i = 1$ to n. FLPE is set when it is required to invert LPP in the parenthesis. FLAB is set when $n>4$. The 3 flags FL2, FLAB, and FLPE give eight possibilities.

(1) when all flags are 0, no action is necessary.

(2) when FL2 and FLAB are 0 and FLPE is 1, no action is necessary, but during standard cell selection NOR cell is selected instead of OR cell.

(3) when FLAB is 1 and FL2 and FLPE are 0, the stored output of each unit in the SYOU array are implemented using OR cell with the proper number inputs. If $n>16$, then more than one OR cells are required as discussed in Section 4.1.

(4) when FLAB and FLPE are 1 and FL2 is 0, the stored output of each unit in the SYOU array is implemented using NOR cell with proper number of inputs. If $n>16$, then the first output of units in SYOU array are implemented using OR cell with the proper number of inputs. The output of these OR cells is stored back into SYOU array. This process is repeated until less than 4 terms are left in the SYOU array.

Table 9: Final Implementation Detail

No.	FL2	FLAB	FLPE		Final implementation
1	0	0	0	$K_i = 1$ for all $i = 1$ to n, $n \leq 4$	No ACTION
2	0	0	0	$K_i = 1$ for all $i = 1$ to n, $n \leq 4$	No ACTION
3 a	0	1	0	$K_i = 1$ for all $i = 1$ to n, $16 \leq n > 4$	OR
b	0	1	1	$K_i = i$ for all $i = 1$ to n, $n > 16$	OR, OR
4 a	0	1	1	$K_i = i$ for all $i = 1$ to n, $16 \leq n > 4$	NOR
b	0	1	1	$K_i = i$ for all $i = 1$ to n, $n > 16$	OR, NOR
5	1	0	0	$K_i = 2$ for any $i = 1$ to n, $n \leq 4$	INVERTER
6	1	0	1	$K_i = 2$ for any $i = 1$ to n, $n \leq 4$	NO ACTION
7 a	1	1	0	$K_i = 2$ for any $i = 1$ to n, $10 \geq 4$	NAND
b	1	1	0	$K_i = 2$ for any $i = 1$ to n, $n > 16$	NAND, OR
8 a	1	1	0	$K_i = 2$ for any $i = 1$ to n, $16 \leq n > 4$	AND
b	1	1	1	$K_i = 2$ for any $i = 1$ to n, $64 \leq n > 16$	NAND, NOR
c	1	1	1	$K_i = 2$ for any $i = 1$ to n, $n > 64$	NAND, OR, NOR

Then NOR cells with the proper number of inputs is provided.

(5) When FL2 is 1 and FLAB and FLPE are 0, the output of the selected standard cell must be inverted.

(6) When FL2 and FLPE are 1, and FLAB is 0, no action is necessary.

(7) When FL2 and FLAB are 1 and FLPE is 0, the NOR cells are selected instead of the OR cell during the standard cell selection to implement units. The stored output of each unit in the SYOU array is implemented using NAND. If n>16, then more than one NAND cells are required. The output of NAND cells are implemented using OR cell.

(8) When all flags are 1, the NOR cells are selected instead of the OR cells during the standard cell selection to implement units. The output of each unit of the SYOU array is implemented using AND cells. If n>16, then implement them using NAND cells. Store the output of each NAND cell in SYOU array. Now it reduces to case 4 discussed above.

If the equation is an RTE, then the output of condition part is used as the clock input to the D-flip-flop (standard cell 1830).

5.3 SUMMARY

The implementation details of the synthesis algorithm were provided in this chapter. The output of the DDLSYN is of the format required by the other CADAT programs.

CHAPTER 6. EXAMPLES

This chapter provides five examples of digital circuit design using DDL description. The examples range from small synchronous circuits to a simple, but complete minicomputer.

6.1 SEQUENTIAL CIRCUIT

Figure 18 shows a sequential circuit along with the DDL description, DDLTRN output, simulation commands, simulation output, synthesis output, and the circuit using the synthesis output. The X(1:5) and S are inputs to the circuit, M(2:5) are the outputs of the circuit. When clock P is present X(2:5) are transferred to modules 1 to 4, M(2) is transferred to module 5 and M(4) is transferred to module 6. A detailed description of Figure 18(c) follows:

Line 1: The name of the system is HELOGIC.

Line 2: Input terminals X, 5 bits long (numbered 1 through 5), and S, 1 bit long.

Line 3: Output terminals M23, M24, and M234, 1 bit long, and M, 4 bits long (numbered 2 through 5).

Line 4: A single phase clock (time) P.

Line 5: A latch by name SW

Line 6: Register A, 6 bits long (numbered 2 through 7). Modules 1 to 6 of Figure 18(a) are named as A(2:7).

Lines 7-9: Declare BOolean declarations corresponding to Figure 18(b).

Lines 10-13: Declare the identifiers used in the above BOolean equations (Line 7-9).

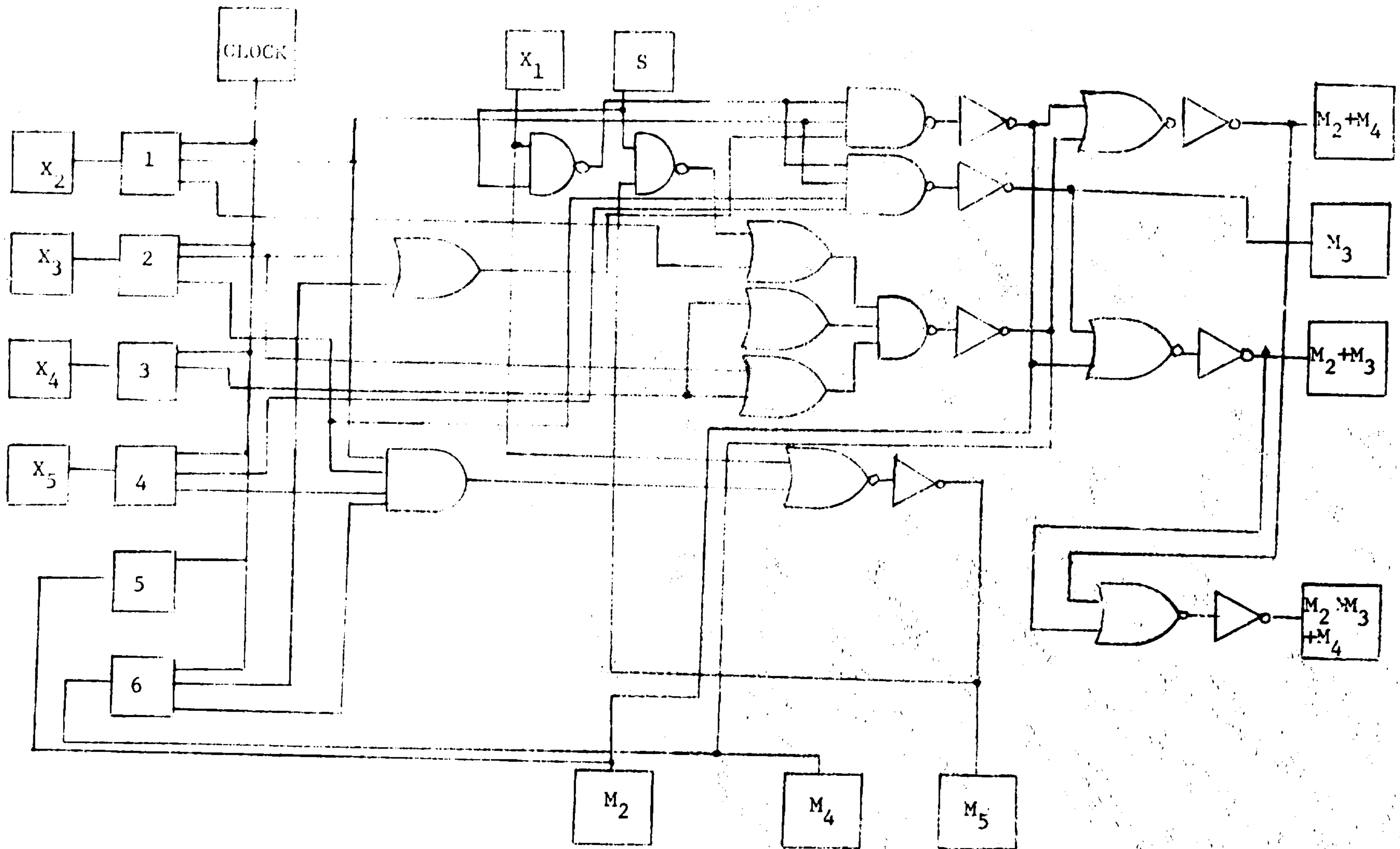


Figure 18(a) - Sequential Circuit

$M_2 = X_3 * M_4' * X_2' * (\overline{X_1 * S_1})$

$M_3 = (\overline{X_1 * S_1}) * X_2' * \overline{X}_3' * X_5'$

$M_4 = (M_5 * S_1 + \overline{Y}_2') * (X_1' + M_2') * (X_3' + X_4')$

$M_5 = X_1 + (X_2' * \overline{X}_3' * \overline{X}_5' * \overline{M}_4')$

Figure 18 (b). Boolean Equations for the Sequential Circuit

```
DIGITAL DESIGN LANGUAGE TRANSLATOR      VERSION - 04.081077      DAY

<SY>HFLOGIC:
<IN>X(5),S.
<TE>N23,N24,N234,N(2:5).
<TI>P.
<LA>SA.
<RE>A(2:7).
<BO>N23=N(2)+N(3),N24=N(2)+N(4),N234=N(2)+N(3)+N(4).
<BO>N(3)=(CC2)*(CC3).
<BO>N(2)=(CC1)*(CC2),N(4)=(CC4)*(CC5)*(CC6),N(5)=X(1)+(CC7).
<ID>CC7=A(2)*TA(3)*TA(5)*TA(7).
<ID>CC1=(A(3)*A(7)*A(2)),CC2=T(X(1)*S),CC3=A(2)*TA(3)*A(5),
      CC4=((X(5)*S)+TA(2)),
      CC5=(A(4)+A(6)),CC6=(A(3)+A(4)).
<AU>CONF:P:
   ISA)A(2:5)<-X(2:5),A(6)<-N(2),A(7)<-N(4)...
<FL>3,4,5,H,H.
```

Figure 18(c): Description for Sequential Circuit

```
PASS1--FACILITIES IDENTIFIED


DECLARED FACILITIES

        <SY> RELOGIC
        <IR> X(1:5)
             S(1:1)
             R23(1:1)
             R24(1:1)
             R234(1:1)
             R(2:5)
        <TI> F(1:1)
        <LA> SW(1:1)
        <RF> A(2:7)
        <JI> CC7(1:1)
             CC1(1:1)
             CC2(1:1)
             CC3(1:1)
             CC4(1:1)
             CC5(1:1)
             CC6(1:1)
             <AU> COMP

DECLARED OPERATIONS

<SY> RELOGIC:
     <RO> R23=R(2) + R(3), R24=R(2) + R(4), R234=R(2) + R(3) + R(4).

     <RO> R(3)=(CC2)*(CC3).

     <RO> R(2)=(CC1)*(CC2), R(4)=(CC4)*(CC5)*(CC6), R(5)=X(1) + (CC7).
     <AU> COMP: F:
          JSR) A(2: 5)<=X(2: 5), A(6)<=R(2), A(7)<=R(4)...
```

Figure 18 (c): (Continued)

```
DIGITAL DESIGN LANGUAGE TRANSLATOR      VERSION - 04.081077      DAY

     PASS2--SYNTAX RECORD


<SY> MELOGIC: M23=M(2) + M(3), M24=M(2) + M(4), M234=M(2) + M(3) + M(4), M(3)=(↑(X(1)*S))*(A(2)*↑A(3)*A(5)),
     M(2)=(A(3)*A(7)*A(2))*(↑(X(1)*S)), M(4)=((M(5)*S) + ↑A(2))*(A(4) + A(6))*(A(3) + A(4)), M(5)=X(1)
     + (A(2)*↑A(3)*↑A(5)*↑A(7)),
     <AU> COMP: F:
          |SM| A(2:5)<-X(2:5), A(6)<-M(2), A(7)<-M(4),... .
```

Figure 18(c): (Continued)

```
DIGITAL DESIGN LANGUAGE TRANSLATOR      VERSION - 04.081077      DAY

     PASS3--CONDITIONS DISTRIBUTED


<SY> MELOGIC:
      N23=(N(2) + N(3)),
     N24=(N(2) + N(4)),
     N234=(N(2) + N(3) + N(4)),
     N(3)=(↑(X(1)*S))*(A(2)*↑A(3)*A(5)),
     N(2)=(A(3)*A(7)*A(2))*(↑(X(1)*S)),
     N(4)=((N(5)*S) + ↑A(2))*(A(4) + A(6))*(A(3) + A(4)),
     M(5)=(X(1) + (A(2)*↑A(3)*↑A(5)*↑A(7))),
     ]F*SN] A(2:5)<-SN*X(2:5).,
     ]F*SN] A(6)<-SN*N(2).,
     ]F*SN] A(7)<-SN*N(4).,
     . .
     PASS4--CONCATENATION


<SY> MELOGIC:
     N23=(N(2) + N(3)),
     N24=(N(2) + N(4)),
     N234=(N(2) + N(3) + N(4)),
     N(3)=(↑(X(1)*S))*(A(2)*↑A(3)*A(5)),
     N(2)=(A(3)*A(7)*A(2))*(↑(X(1)*S)),
     N(4)=((N(5)*S) + ↑A(2))*(A(4) + A(6))*(A(3) + A(4)),
     N(5)=(X(1) + (A(2)*↑A(3)*↑A(5)*↑A(7))),
     ]F*SN] A(2:5)<-SN*X(2:5).,
     ]F*SN] A(6)<-SN*N(2).,
     ]F*SN] A(7)<-SN*N(4).,
     .
```

Figure 18(c) (Continued)

```
PASS5--OPERATIONS GATHERED

<SY> RELOGIC:
     N23=(N(2) + N(3)),
     N24=(N(2) + N(4)),
     N234=(N(2) + N(3) + N(4)),
     N(3)=(T(X(1)*S))*(A(2)*TA(3)*A(5)),
     N(2)=(A(3)*A(7)*A(2))*(T(X(1)*S)),
     N(4)=((N(5)*S) + TA(2))*(A(4) + A(6))*(A(3) + A(4)),
     N(5)=(X(1) + (A(2)*TA(3)*TA(5)*TA(7))),
     ]P*SV] A(2:5)<-SA*X(2:5)..
     ]P*SV] A(6)<-SA*N(2)..
     ]P*SV] A(7)<-SA*N(4)..
     .
     PASS6--SUBFACILITIES DISJOINED

<SY> RELOGIC:
     N23=(N(2) + N(3)),
     N24=(N(2) + N(4)),
     N234=(N(2) + N(3) + N(4)),
         N(2)=(A(3)*A(7)*A(2))*(T(X(1)*S)),
     N(3)=(T(X(1)*S))*(A(2)*TA(3)*A(5)),
     N(4)=((N(5)*S) + TA(2))*(A(4) + A(6))*(A(3) + A(4)),
     N(5)=(X(1) + (A(2)*TA(3)*TA(5)*TA(7))),

     ]P*SV] A(2:5)<-SA*X(2:5)..
     ]P*SV] A(6)<-SA*N(2)..
     ]P*SV] A(7)<-SA*N(4)..
     .
```

Figure 18(c): (Continued)

FACILITY TABLE

1	FFLOGIC	1	1	1	-1	0	4	0	0
2	X	1	5	5	-9	0	0	0	0
3	S	1	1	1	-9	0	0	0	0
4	M23	1	1	1	-9	0	11	0	24
5	M24	1	1	1	-9	0	25	0	38
6	M234	1	1	1	-9	0	39	0	58
7	M	2	5	4	-9	0	0	0	0
8	P	1	1	1	-5	0	0	0	0
9	SA	1	1	1	-8	0	0	0	0
10	A	2	7	6	-6	0	0	0	0
11	A	7	7	1	10	0	534	537	532
12	A	6	6	1	10	0	548	551	546
13	A	2	5	4	10	0	564	567	562
14		0	0	0	0	0	0	0	0
15		0	0	0	0	0	0	0	0
16		0	0	0	0	0	0	0	0
17		0	0	0	0	0	0	0	0
18	COMP	1	1	1	-6	0	0	0	0
19	M	5	5	1	7	0	419	0	442
20	M	4	4	1	7	0	444	0	474
21	M	3	3	1	7	0	476	0	499
22	M	2	2	1	7	0	501	0	523

Figure 18(c): (Continued)

```
DESIGN LANGUAGE SIMULATOR      VERSION MSFC 1979      SIMULATION RUN  1      PAGE  1

<FL>3,6
<IN>SW/1
<TR>IR/TP/
<RE>IR/S,X/0,23,0,27,1,10,1,14,1,33,0,23,0,37,1,04,0,33,1,35,0,02,
1,14,1,10,1,13
<OU>IR/COMP,A,S,X,R,R24,R23,R234/
<SI>
```

Figure 18(d): Simulation Commands

TIME	CCLK	A	S	X	M	M24	M25	Z54
0	0	000000	0	00000	0000	0	0	0
2	0	001100	0	10011	0011	1	0	1
4	0	011101	0	10111	0011	1	0	1
6	0	100001	1	01000	0000	0	0	0
8	0	110000	1	01100	0000	0	0	0
10	0	101100	1	11011	0011	1	0	1
12	0	001100	0	10011	0011	1	0	1
14	0	111101	0	11111	1001	1	1	1
16	0	010010	1	00100	0010	1	0	1
18	0	101101	0	11011	0101	0	1	1
20	0	110101	1	11101	0001	0	0	0
22	0	001010	0	00010	0010	1	0	1
24	0	110001	1	01100	1000	1	1	1
26	0	100010	1	01000	0001	0	0	0
28	0	101100	1	01011	0100	0	1	1

END OF FILE REACHED ON INPUT
SIMULATION TERMINATED AT TIME = 29

Figure 18(e): Simulation Output

Note that identifier names are not required to be declared in the terminal declaration. (For example, CC1 is not required to be declared as terminal). Also, sometimes using <ID> rather than <BO> reduces the circuit. For example, if CC1 was declared as BOolean, then it would require 4 standard cells to implement M(2) (an inverter and a 2-input AND cells to implement CC2, a 3-input AND to implement CC1, a 2-input AND to implement outputs of CC1 and CC2). The current description requires 3 standard cells to implement M(2) (an inverter and a 2-input AND cells to implement CC2, a 4-input AND to implement CC1 and output of CC2).

Line 14: Automaton CONP is controlled by clock P.

Line 15: When SW is 1, (2:5) is transferred to A(2:5), M(2) is transferred to A(7), if clock P is present. The periods at the end of line 15 terminate the IF-THEN on SW; AU and SY declarations respectively.

Line 16: Sets the flags of DDLTRN to output the results of each of the six phases and the facility table (see Table 2).

The first 7 equations of the DDLTRN phase 6 output are BEs and the rest of them are RTEs. Figure 18(d) shows the input commands for DDLSIM. Flags for DDLSIM are set for octal data input (3) and binary output (6) in Line 1 (refer to Table 4.2 of [4]). SW is initialized to 1 on Line 2. An input/output trigger TR (raising edge of P) is declared in Line 3. Lines 4 and Line 5 read in the values for S and X when TR is on. (There are 14 sets of values). The values of CONP, A, S, X, M, M23, M24 are to be output when TR is on (Line 6). The simulation is started with <SI> in Line 7. Figure 18(e) shows the simulation output.

IDENTIFIER TABLE

NO.	IDENTIFIER	NO.	IDENTIFIER	NO.	IDENTIFIER	NO.	IDENTIFIER
1	M23(1)	2	M(2)	3	M(3)	4	M24(1)
5	M(4)	6	M234(1)	7	A(3)	8	A(7)
9	A(2)	10	X(1)	11	S(1)	12	XXXXXXXX(1)
13	XXXXXXXX(1)	14	A(5)	15	XXXXXXXX(1)	16	XXXXXXXX(1)
17	XXXXXXXX(1)	18	M(5)	19	A(4)	20	A(6)
21	XXXXXXXX(1)	22	XXXXXXXX(1)	23	XXXXXXXX(1)	24	XXXXXXXX(1)
25	XXXXXXXX(1)	26	XXXXXXXX(1)	27	XXXXXXXX(1)	28	XXXXXXXX(1)
29	XXXXXXXX(1)	30	XXXXXXXX(1)	31	F(1)	32	S7(1)
33	X(2)	34	XXXXXXXX(1)	35	XXXXXXXX(1)	36	X(5)
37	XXXXXXXX(1)	38	X(4)	39	XXXXXXXX(1)	40	X(5)
41	XXXXXXXX(1)	42	XXXXXXXX(1)	43	XXXXXXXX(1)	44	XXXXXXXX(1)
45	XXXXXXXX(1)						

CELL TABLE

CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL
1	1720	2	1720	3	1730	4	1220	5	1630	6	[illegible]	7	[illegible]
8	1230	9	1220	10	1630	11	1720	12	1720	13	[illegible]	14	[illegible]
15	1720	16	1310	17	1630	18	1310	19	1310	20	1310	21	1640
22	1720	23	1620	24	1620	25	1630	26	1620	27	1630	28	1620
29	1830	30	1620	31	1830	32	1620	33	1620	34	1830	35	1620
36	1620	37	1830										

Figure 18(f): Synthesis Output

NET TABLE

NET	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	F.
1	1	4																		
2	1	3	2	3	3	4	6	4	6	5	33	2								
3	1	2	3	3	10	4	10	5												
4	2	4																		
5	2	2	3	2	17	5	36	2												
6	3	5																		
7	5	4	7	2	11	3	16	2	27	4										
8	5	3	20	2	37	4														
9	5	2	6	4	14	2	21	5	25	4										
10	4	3	9	3	22	3														
11	4	2	9	2	13	2														
12	4	4	6	2																
13	5	5	6	3																
14	8	2	19	2	31	4														
15	7	3	8	3																
16	8	5	10	2																
17	9	4	10	3																
18	13	3	22	4																
19	11	2	12	3	29	4														
20	16	2	34	4																
21	11	4	17	2																
22	12	4	17	3																
23	13	4	15	3																
24	14	3	15	2																
25	15	4	16	2	17	4														
26	16	3																		
27	18	3	21	4																
28	19	3	21	3																
29	20	3	21	2																
30	21	4	22	2																
31	23	[illegible]	32	3	35	3														
32	23	2	24	3	26	3	28	3	30	3	32	2	33	3	35	2	36	3		
33	24	2																		
34	23	4	25	2	27	2	29	2	31	2										
35	24	4	25	3																
36	26	2																		
37	26	4	27	3																
38	28	2																		
39	28	4	29	3																
40	30	2																		
41	31	4	31	3																
42	32	4	34	2																
43	33	4	34	3																
44	38	4	37	2																
45	38	4	37	4																

Figure 18(?): (Continued)

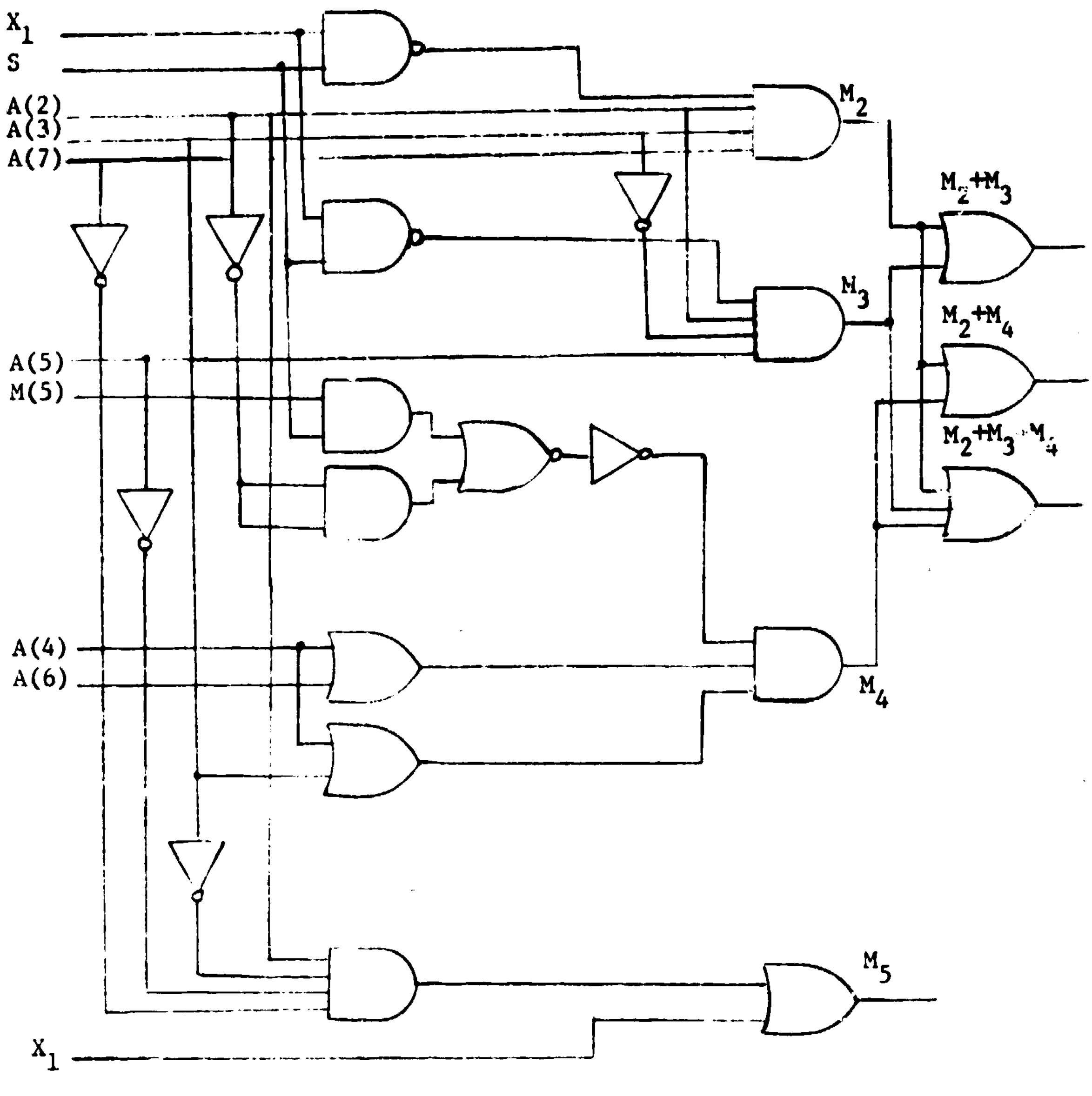


Figure 18(g): Sequential Circuit (From DDLSYN Output)

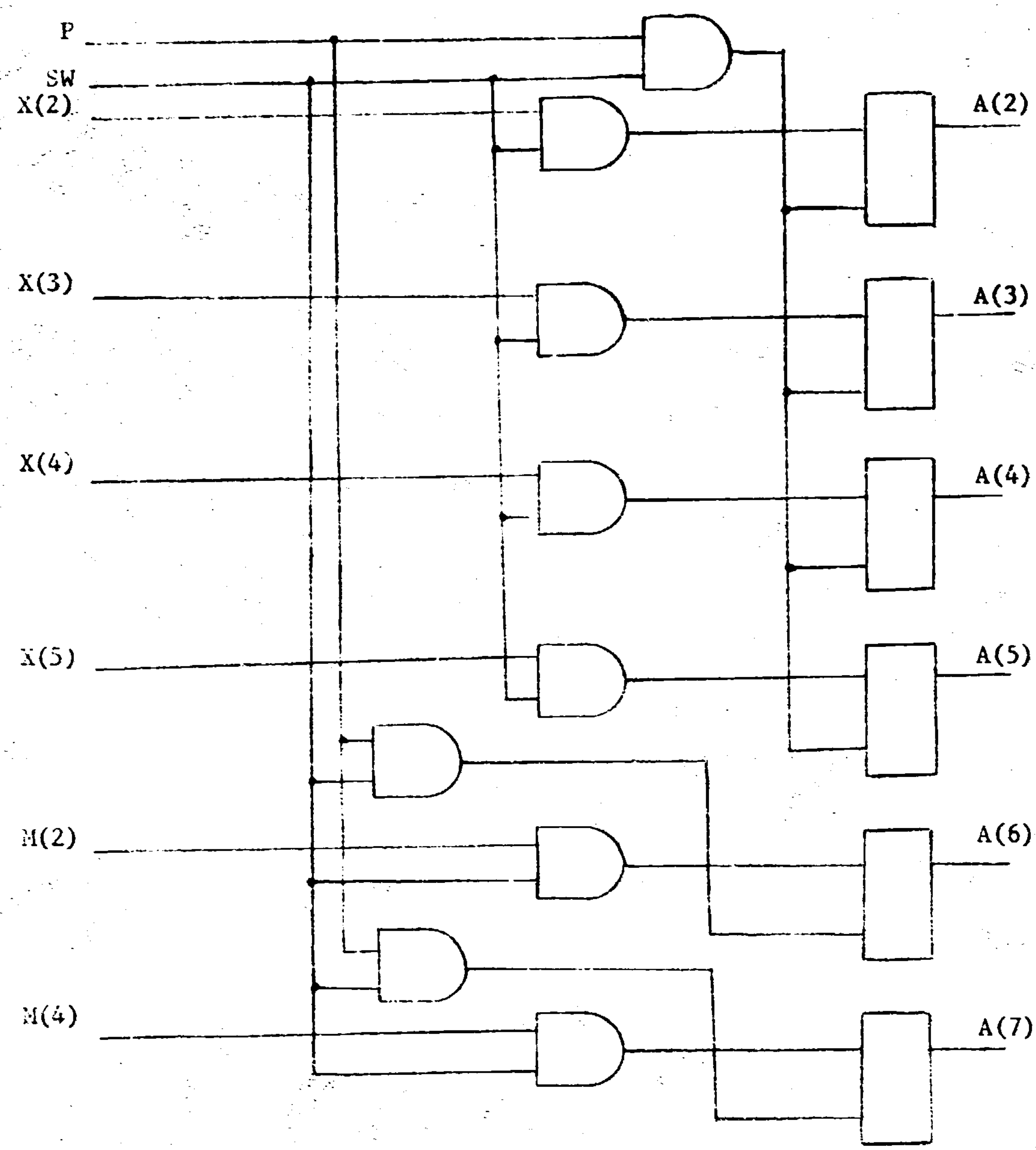


Figure 18(g): (Continued)

Figure 18(f) shows the synthesis output for the sequential circuit. The NET 1 for the identifier M23(1) has only one entry in the NET Table. M23(1) is connected to pin number 4 of cell number 1 (cell number 1 corresponds to the standard cell 1720 in the cell table).

Figure 18(g) shows the circuit diagram for the sequential circuit drawn from synthesis output. There are more cells in the automatic design compared to the manual design. The excessive cells are due to the following:

(1) ^(X(1)*S) is repeated in equations M(2) and M(3). This repetition adds 2 more cells (an inverter and a 2-input AND). This can be avoided by declaring (X(1)*S) as:

<BO> Z = ^(X(1)*S) and using Z in equations M(2) and M(3).

(2) The SW test condition is not necessary in Line 15 of Figure 18(c). The SW condition of the BEs and RTEs in the DDLTRN output adds 9, 2-input AND cells. Without SW condition in an Automaton the simulator gives an error condition.

(3) An inverted output is available for the D-Flip-flop. For ^A(2), ^A(3), ^A(5), and ^A(7) use of the inverted output can save 4 - inverters.

(1) and (2) above are DDL description dependent, (3) can not be avoided in the present implementation because the registers are implemented later in the process and as such, the inverted output information is not available during BE synthesis. The manual design is drawn with single gate cells only, whereas the automatic design uses larger cells.

6.2 SERIAL TWOS COMPLEMENTER

Figure 19 shows a serial 2's complementer circuit along with the DDL description, DDLTRN output, simulation commands, synthesis output, and the circuit using the synthesis output. For DDL description (Figure 19(b) and simulation commands (Figure 19(c), refer to [4]. Figure 19(d) shows the simulation output. Figure 19(e) shows the synthesis output for the serial 2's complementer design.

Figure 19(f) shows the circuit design for serial 2's complementer drawn from synthesis output. There are more cells in the automatic design compared to the manual design. The excessive cells are due to the following:

(1) An inverter and a 3-input AND cell can be saved by declaring C(2)*^C(1)*C(0) as BOolean declaration.

(2) The SW and T conditions in the declaration are not necessary in the lines 10 and 11 of Figure 19(b). The SW and T conditions of the BEs and RTEs in the DDLTRN output adds 10, 2-input AND cells, and 4 inverters in the automatic design.

(3) As discussed in 6.1, inverters for ^C(1), ^S, ^R(6) can be saved by using available inverted outputs of the D-Flip-flops.

(1) and (2) above are also DDL description dependent, (3) can not be avoided in the present implementation because the registers are implemented later in the process, and as such the inverted output information is not available during BE synthesis.

The state declaration in the automatic design changes automaton COMP to a register and corresponding states BEs. The register transfer occurs only if the condition part (part of which in turn is dependent on BE output corresponding to the state) is satisfied. The manual design

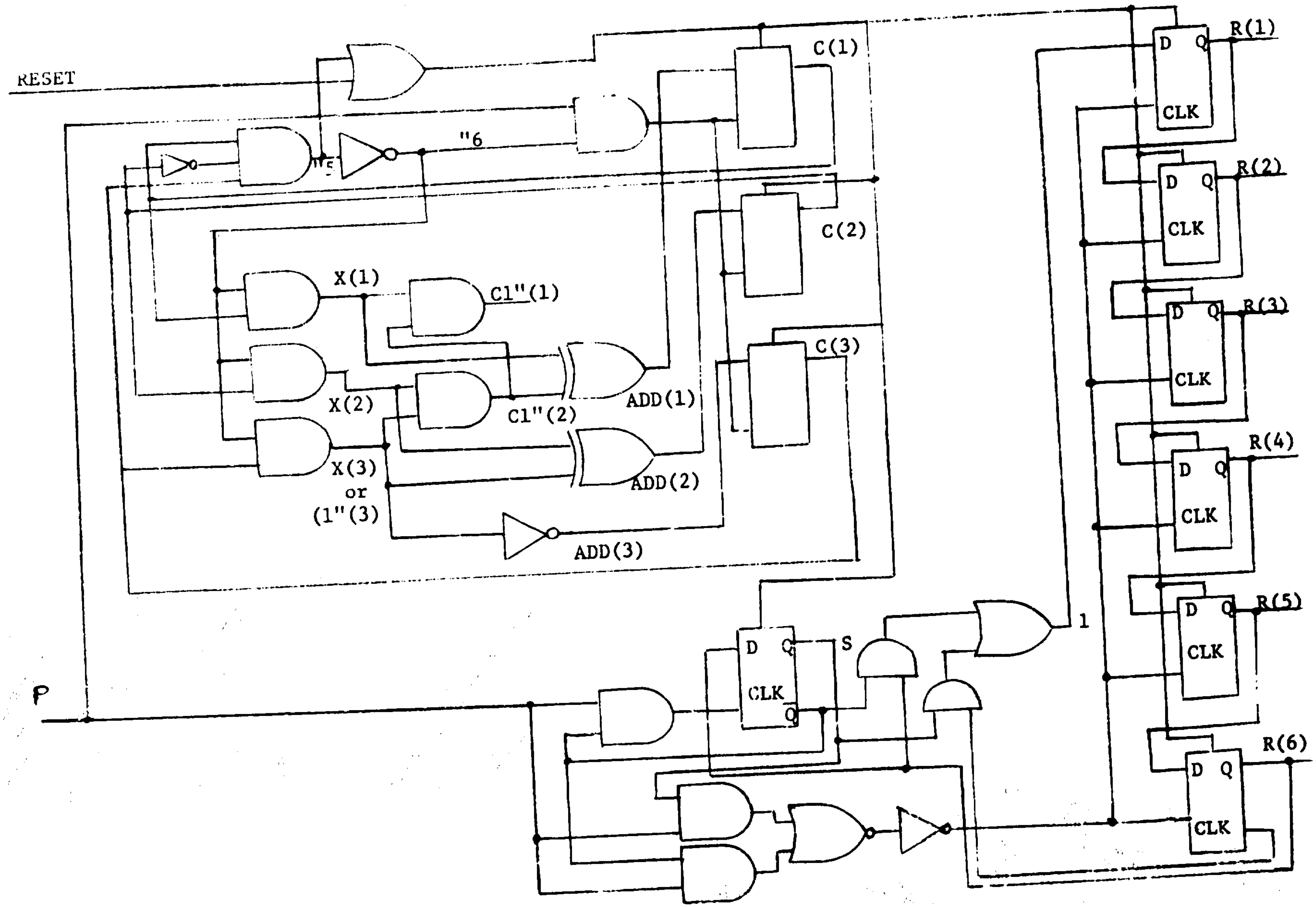


Figure 19(a): Serial 2's Complementer

```
DIGITAL DESIGN LANGUAGE TRANSLATOR      VERSION = 04.081077      DAY

<SY>COMPLEMENTER:
<RE>R(1:6),C(2:0),S,I.
<LA>SA.
<TI>P.
<CP> ADD(3);X;
     <IE> X(3),C(3).
          <IO> CC=(C(2:3)(1,1)).
     <MO> C=X*CC,ADD=X@CC..
<AU>COMP:P:
     <S1>I(0):SA:I<-1,C<-0,S<-0,->S1.
        S1(1):1:)S) R(1)<-1R(6), C(2:0)<- (1:2);S<- (0),R<-R(6)(R(1:5)..
        )C(2)*1C(1)*C(0)]I<-0,->I;C<-ADD:C.....
<FL>3,4,5,6,8.
```

Figure 19(b): Description for Serial 2's Complementer

```
PASS1--FACILITIES IDENTIFIED

DECLARED FACILITIES

      <SY> LEMENTER
      <RE> R(1:6)
           C(2:0)
           S(1:1)
           T(1:1)
      <LA> SW(1:1)
      <TI> F(1:1)
           <OP> ADD(1:3)
           <TE> X(1:3)                                    C -> C"1

                C"1(1:3)
           <ID> CC(1:1)
           <AU> COMP
           <SI> I
                S1


DECLARED OPERATIONS

<SY> LEMENTER:
     <OP> ADD$X$
          <BO> C=X*CC, ADD=X@CC..

     <AU> COMP: F:
          <SI>
               I: SW: T<-1, C<-C, S<-0, ->S1.

               S1: I:
                    )S) R(1)<-TR(F), R(2: 6)<-R(1: 5); S<-R(6), R<-R(6)(R(1: 5)..
                    )C(2)*TC(1)*((0)) I<-0, ->I; C<-ADD*C$.....
```

Figure 19(b): (Continued)

```
DIGITAL DESIGN LANGUAGE TRANSLATOR      VERSION - 04.0M1077      DAY

     PASS2--SYNTAX REDUCED


<SY> LPMENTER: I=*/COMP'0, S1=*/COMP'101  , C"1=X*C"1(2:3)11D1  , ADD=X&C"1(2:3)11D1  ,
     <AU> COMP: P:
          ]1]
               ]SW] I<-101  , C<-0, S<-0, ->S1..,
          ]S1]
               ]1]
                    ]S] R(1)<-TR(6), R(2:6)<-R(1:5); S<-R(6), R<-R(6)[R(1:5).,
                    ]C(2)*TC(1)*C(0)] I<-0, ->1; C<-ADD , X=C...., ., .
```

Figure 19(b): (Continued)

DIGITAL DESIGN LANGUAGE TRANSLATOR

PASS3--CONDITIONS DISTRIBUTED

```
<SY> LEMENTER:
     I=*/COMP'0,
     S1=*/COMP'101   ,
     "1=I*S0,
     "2=S1*1,
     "3="2*S,
     "4="2*TS,
     "5="2*C(2)*TC(1)*C(0),
     "6="2*T(C(2)*TC(1)*C(0)),
      C"1=X*C"1(2:3)(101   ,
     ADD=(X@C"1(2:3)(101   ),
     IF*"1] T<="1*101   ..
     IF*"1] C<="1*0..
     IF*"1] S<="1*0..
     IF*"1] COMP<="1*101   ..
     IF*"3] R(1)<="3*TR(6)..
     IF*"3] R(2:6)<="3*R(1:5)..
     IF*"4] S<="4*R(6)..
     IF*"4] R<="4*R(6)(R(1:5)..
     IF*"5] T<="5*0..
     IF*"5] COMP<="5*0..
     IF*"6] C<="6*ADD..
     X="6*C, .
     .
```

DIGITAL DESIGN LANGUAGE TRANSLATOR

PASS4--CONCATENATION REMOVED

```
<SY> LEMENTER:
     I=*/COMP'0,
     S1=*/COMP'101   ,
     "1=I*S0,
     "2=S1*1,
     "3="2*S,
     "4="2*TS,
     "5="2*C(2)*TC(1)*C(0),
     "6="2*T(C(2)*TC(1)*C(0)),
     C"1(1:2)=X(1:2)*C"1(2:3),
     C"1(3)=X(3)*101   ,
     ADD(1:2)=(X(1:2)@C"1(2:3)),
     ADD(3)=(X(3)@101   ),
     IF*"1] T<="1*101   ..
     IF*"1] C<="1*0..
     IF*"1] S<="1*0..
     IF*"1] COMP<="1*101   ..
     IF*"3] R(1)<="3*TR(6)..
     IF*"3] R(2:6)<="3*R(1:5)..
     IF*"4] S<="4*R(6)..
     IF*"4] R(1)<="4*R(6)..
     IF*"4] R(2:6)<="4*R(1:5)..
     IF*"5] T<="5*0..
     IF*"5] COMP<="5*0..
     IF*"6] C<="6*ADD..
     X="6*C, .
```

Figure 19(b): (Continued)

PASS5--OPERATIONS GATHERED

```
<SY> LEMENTER:
     I=*/COMP'0,
     S1=*/COMP'101  ,
     "1=I*SW,
     "2=S1*I,
     "3="2*S,
     "4="2*↑S,
     "5="2*C(2)*↑C(1)*C(0),
     "6="2*↑(C(2)*↑C(1)*C(0)),
     C"1(1:2)=X(1:2)*C"1(2:3),
     C"1(3)=X(3)*101  ,
     ADD(1:2)=(X(1:2)aC"1(2:3)),
     ADD(3)=(X(3)a101  ),
     ]P*"1 + P*"5] T<-"1*101    + "5*0.,
     ]P*"1 + P*"6] C<-"1*0 + "6*ADD.,
     ]P*"1 + P*"4] S<-"1*0 + "4*F(6).,
     ]P*"1 + P*"5] COMP<-"1*101    + "5*0.,
     ]P*"3 + P*"4] R(1)<-"3*↑R(6) + "4*R(6).,
     ]P*"3 + P*"4] R(2:6)<-"3*R(1:5) + "4*R(1:5).,
     X="6*C, .
```

PASS6--SUBFACILITIES DISJOINED

```
<SY> LEMENTER:
     I=*/COMP'0,
     S1=*/COMP'101  ,
     "1=I*SW.
     "2=S1*I,
     "3="2*S,
     "4="2*↑S,
     "5="2*C(2)*↑C(1)*C(0),
     "6="2*↑(C(2)*↑C(1)*C(0)),
       C"1(1:2)=X(1:2)*C"1(2:3),
     C"1(3)=X(3)*101  ,
       ADD(1:2)=(X(1:2)aC"1(2:3)),
     ADD(3)=(X(3)a101  ),
     ]P*"1 + P*"5] T<-"1*101    + "5*0.,
     ]P*"1 + P*"6] C<-"1*0 + "6*ADD.,
     ]P*"1 + P*"4] S<-"1*0 + "4*F(6).,
     ]P*"1 + P*"5] COMP<-"1*101    + "5*0.,

     ]P*"3 + P*"4] R(1)<-"3*↑R(6) + "4*R(6).,
     ]P*"3 + P*"4] R(2:6)<-"3*R(1:5) + "4*R(1:5).,
     X="6*C, .
```

Figure 19(b): (Continued)

FACILITY TABLE

1	[illegible]	1	1	1	-1	0	439	0	0
2	F	1	6	6	-6	0	0	0	0
3	C	2	0	-3	-6	0	227	358	362
4	S	1	1	1	-6	0	235	291	295
5	I	1	1	1	-6	0	216	321	325
6	SR	1	1	1	-8	0	0	0	0
7	F	1	1	1	-5	0	0	0	0
8	ADD	1	3	3	-9	0	0	0	0
9	X	1	3	3	-9	0	191	0	194
10	M1	1	1	1	-9	0	213	0	219
11	CM1	1	3	3	-9	0	0	0	0
12	M2	1	1	1	-9	0	252	0	258
13	COMP	1	1	1	-6	0	244	330	340
14	I	17	0	1	-13	0	196	0	201
15	SI	16	1	1	-13	0	203	0	208
16	1C1		1	1	-17	0	0	0	0
17	O		0	1	-17	0	0	0	0
18	M3	1	1	1	-9	0	255	0	264
19	M4	1	1	1	-9	0	280	0	287
20	M5	1	1	1	-9	0	304	0	318
21	M6	1	1	1	-9	0	337	0	354
22		0	0	0	0	0	0	0	0
23		0	0	0	0	0	0	0	0
24	CM1	3	3	1	11	0	513	0	519
25	CM1	1	2	2	11	0	521	0	531
26	ALU	3	3	1	8	0	489	0	497
27	ALL	1	2	2	8	0	499	0	511
28	M	2	6	5	2	0	456	463	454
29	M	1	1	1	2	0	480	487	478

Fibure 19(b): (Continued)

```
DIGITAL DESIGN LANGUAGE SIMULATOR      VERSION MSFC 1979

        1:      <FL>4,6
        2:      <IN>SR/1
        3:      <RE>T/R/5,20
        4:      <TR>OUTTR/TR/
        5:      <CO>OUTTR/COMP,R,S,C,T/,T/R/
        6:      <SI>
```

Figure 19(c): Simulation Commands

DIGITAL DESIGN LANGUAGE SIMULATOR   VERSION MSFC 1979

TIME	CLK	R	S	C	T	R
0	0	000000	0	000	0	000000
2	1	000000	0	000	1	
4	1	100010	1	001	1	
6	1	110001	1	010	1	
8	1	011000	1	011	1	
10	1	101100	1	100	1	
12	1	110110	1	101	1	
14	0	111011	1	101	0	111011
16	1	111011	0	000	1	
18	1	001010	0	001	1	
20	1	000101	0	010	1	
22	1	100010	1	011	1	
24	1	110001	1	100	1	
26	1	011000	1	101	1	
28	0	101100	1	101	0	101100
30	1	101100	0	000	1	

END OF FILE REACHED ON INPUT
SIMULATION TERMINATED AT TIME = 31

Figure 19(d): Simulation Output

IDENTIFIER TABLE

NO.	IDENTIFIER	NO.	IDENTIFIER	NO.	IDENTIFIER	NO.	IDENTIFIER
1	I(1)	2	COMP(1)	3	SI(1)	4	"I(1)
5	SK(1)	6	"2(1)	7	I(1)	8	"3(1)
9	S(1)	10	"4(1)	11	XXXXXXXX(1)	12	"5(1)
13	C(2)	14	C(1)	15	C(0)	16	XXXXXXXX(1)
17	"6(1)	18	XXXXXXXX(1)	19	XXXXXXXX(1)	20	C"1(1)
21	R(1)	22	C"1(2)	23	R(2)	24	C"1(3)
25	X(3)	26	ADD(1)	27	ADD(2)	28	ADD(3)
29	P(1)	30	XXXXXXXX(1)	31	XXXXXXXX(1)	32	XXXXXXXX(1)
33	XXXXXXXX(1)	34	XXXXXXXX(1)	35	XXXXXXXX(1)	36	XXXXXXXX(1)
37	R(6)	38	XXXXXXXX(1)	39	XXXXXXXX(1)	40	XXXXXXXX(1)
41	XXXXXXXX(1)	42	XXXXXXXX(1)	43	R(1)	44	XXXXXXXX(1)
45	XXXXXXXX(1)	46	XXXXXXXX(1)	47	XXXXXXXX(1)	48	XXXXXXXX(1)
49	XXXXXXXX(1)	50	R(2)	51	XXXXXXXX(1)	52	XXXXXXXX(1)
53	XXXXXXXX(1)	54	XXXXXXXX(1)	55	R(3)	56	XXXXXXXX(1)
57	XXXXXXXX(1)	58	R(4)	59	XXXXXXXX(1)	60	XXXXXXXX(1)
61	Y(5)	62	XXXXXXXX(1)	63	XXXXXXXX(1)	64	XXXXXXXX(1)
65	XXXXXXXX(1)	66	XXXXXXXX(1)				

Figure 19(e): Synthesis Output

CELL TABLE

CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL
1	1310	2	1620	3	1620	4	1620	5	1310	6	1620	7	1310
8	1640	9	1310	10	1230	11	1620	12	1620	13	1620	14	2310
15	2310	16	1310	17	1870	18	1310	19	1830	20	1870	21	1310
22	1620	23	1830	24	1620	25	1830	26	1620	27	1830	28	1870
29	1310	30	1620	31	1830	32	1870	33	1310	34	1830	35	1870
36	1390	37	1310	38	1310	39	1870	40	1310	41	1830	42	1870
43	1310	44	1870	45	1310	46	1830	47	1870	48	1310	49	1830
50	1870	51	1310	52	1830	53	1870	54	1300	55	1310	56	1830
57	1870	58	1310	59	1830	60	1620	61	1620	62	1620		

Figure 19(e): (Continued)

ET TABLE

NET	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	F
1	1	3	2	3																		
2																						
3	3	3	1	2																		
4	2	4	17	4	19	3	20	4	28	4	32	4	34	3								
5	2	2																				
6	3	4	4	3	6	3	8	5	11	3												
7	3	2	19	4																		
8	4	4	35	4	39	5	42	4	44	5	47	5	50	5	54	5	57	5				
9	4	2	5	2	31	4																
10	8	4	28	2	30	3	35	2	39	3	42	2	44	3	47	3	50	3	54			
11	5	3	6	2																		
12	6	6	17	2	32	2																
13	8	4	10	4	23	4	60	2														
14	7	2	9	2	25	4	61	2														
15	8	2	10	2	27	4	62	2														
16	7	3	8	3																		
17	11	4	20	2	22	3	24	3	26	3	60	3	61	3	62	3						
18	9	3	10	3																		
19	10	5	11	2																		
20	12	4																				
21	12	3	14	2	60	4																
22	12	2	13	4	14	3																
23	13	3	15	2	61	4																
24	13	2	15	3	16	2																
25																						
26	14	4	22	2																		
27	15	4	24	2																		
28	16	3	26	2																		
29	17	3	17	5	20	3	20	5	[illegible]	3	28	5	32	3	32	5	35	3	36			
30	17	6	18	2																		
31	18	3	19	2																		
32	20	6	21	2																		
33	21	3	23	2	25	2	27	2														
34	22	4	23	3																		
35	24	4	25	3																		
36	26	4	27	3																		
37	30	2	38	2	39	2	59	4														
38	28	8	29	2																		
39	29	3	31	2																		
40	30	4	31	3																		
41	32	6	33	2																		
42	33	3	34	2																		
43	41	4	44	2	44	4																
44	36	3	36	5	42	3	42	5														
45	38	8	37	2																		
46	37	3	41	2																		
47	38	3	39	4																		
48	39	8	40	2																		
49	40	3	41	3																		
50	46	4	47	2	[illegible]	4																
51	42	6	43	2																		
52	43	3	46	2	49	2	52	2	56	2	59	2										
53	44	8	45	2																		

Figure 19(e): (Continued)

54	45	3	46	3		
55	49	4	50	2	50	4
56	47	6	48	2		
57	48	3	49	3		
58	52	4	53	2	54	4
59	50	6	51	2		
60	51	3	52	3		
61	56	4	57	2	57	4
62	54	3	54	3	57	3
63	54	6	55	2		
64	55	3	56	3		
65	57	6	58	2		
66	58	3	59	3		

Figure 19(e): (Continued)

C-2

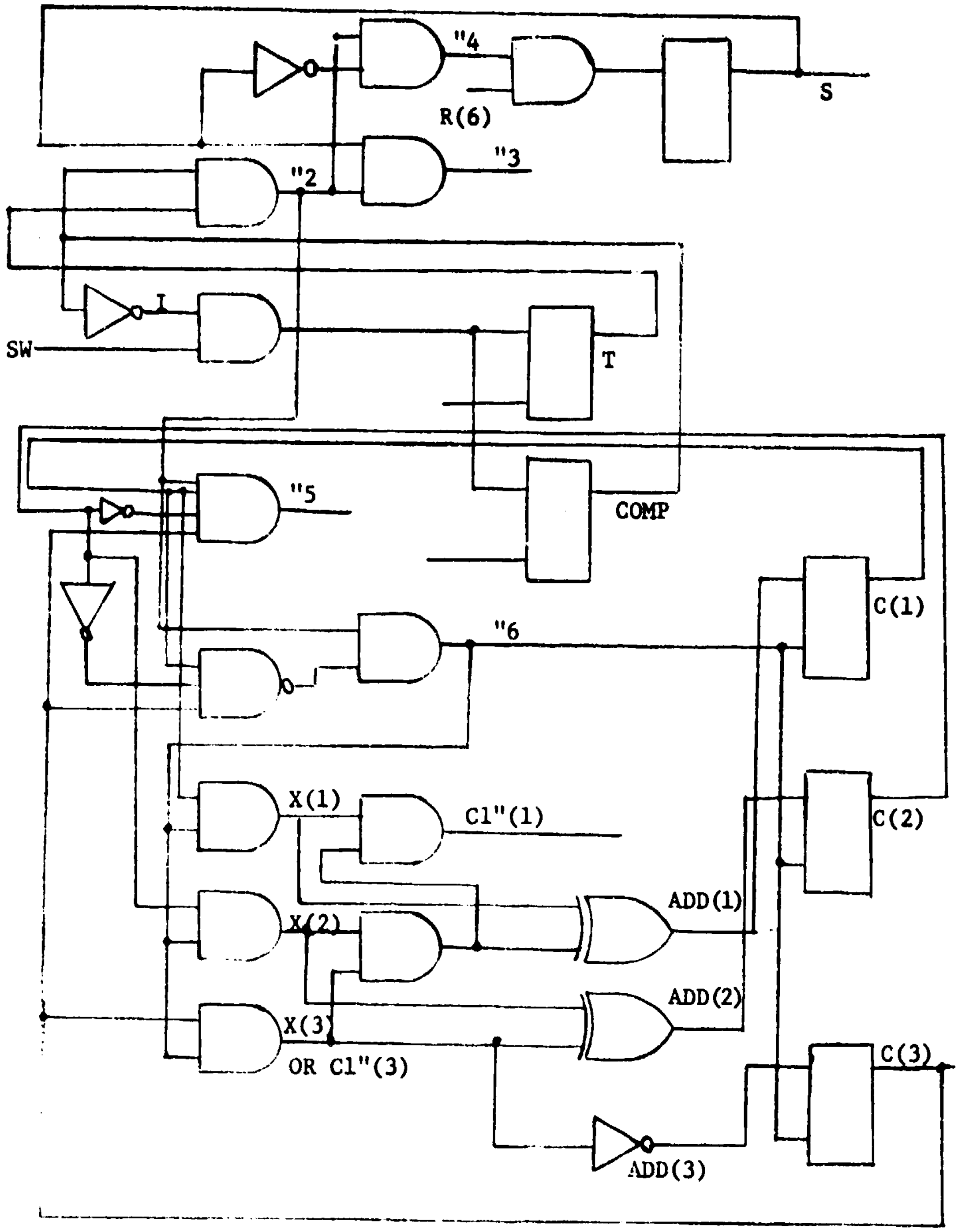


Figure 19(f): Serial 2's Complementer (from DDLSYN output)

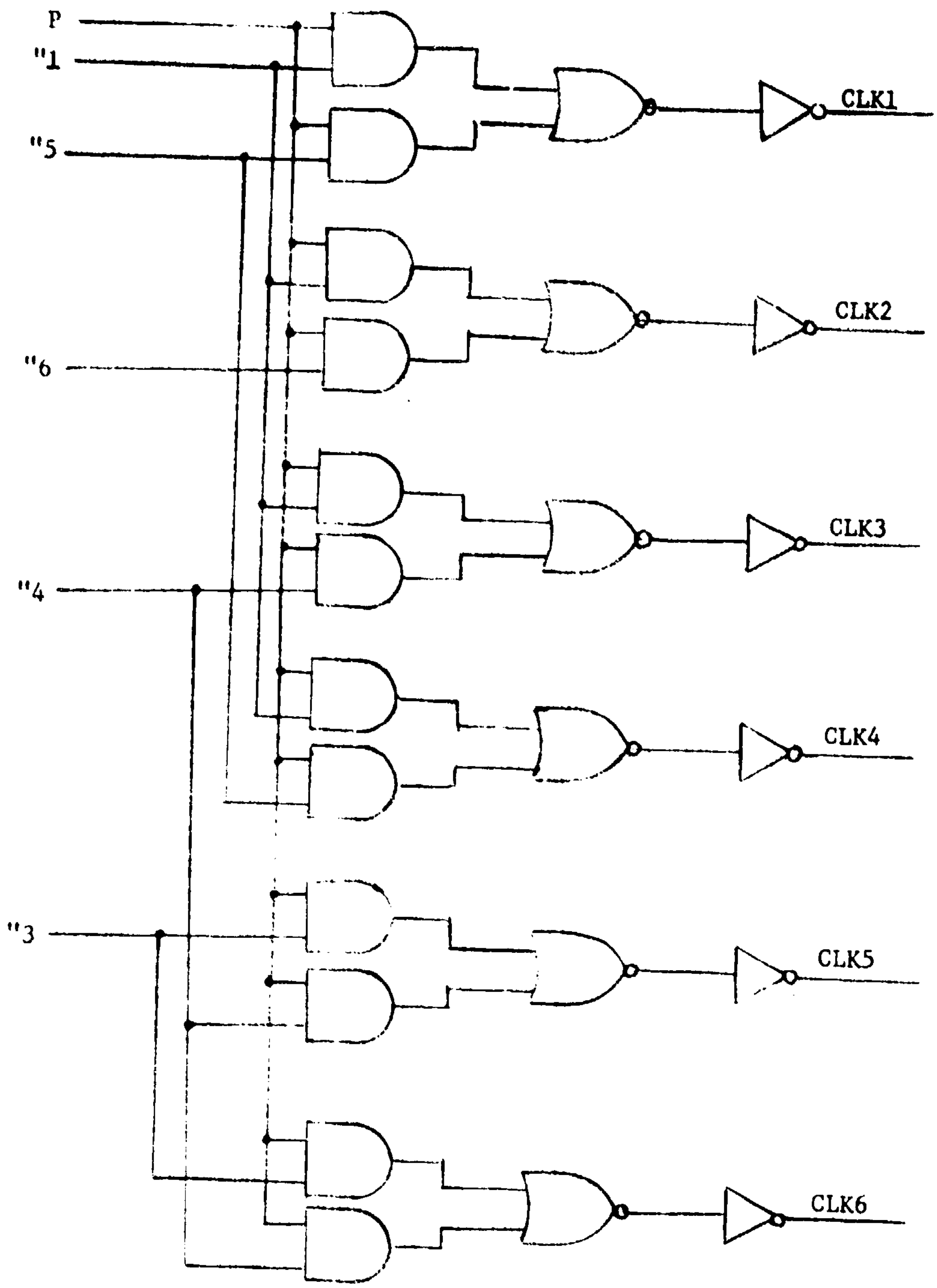


Figure 19(f): (Continued)

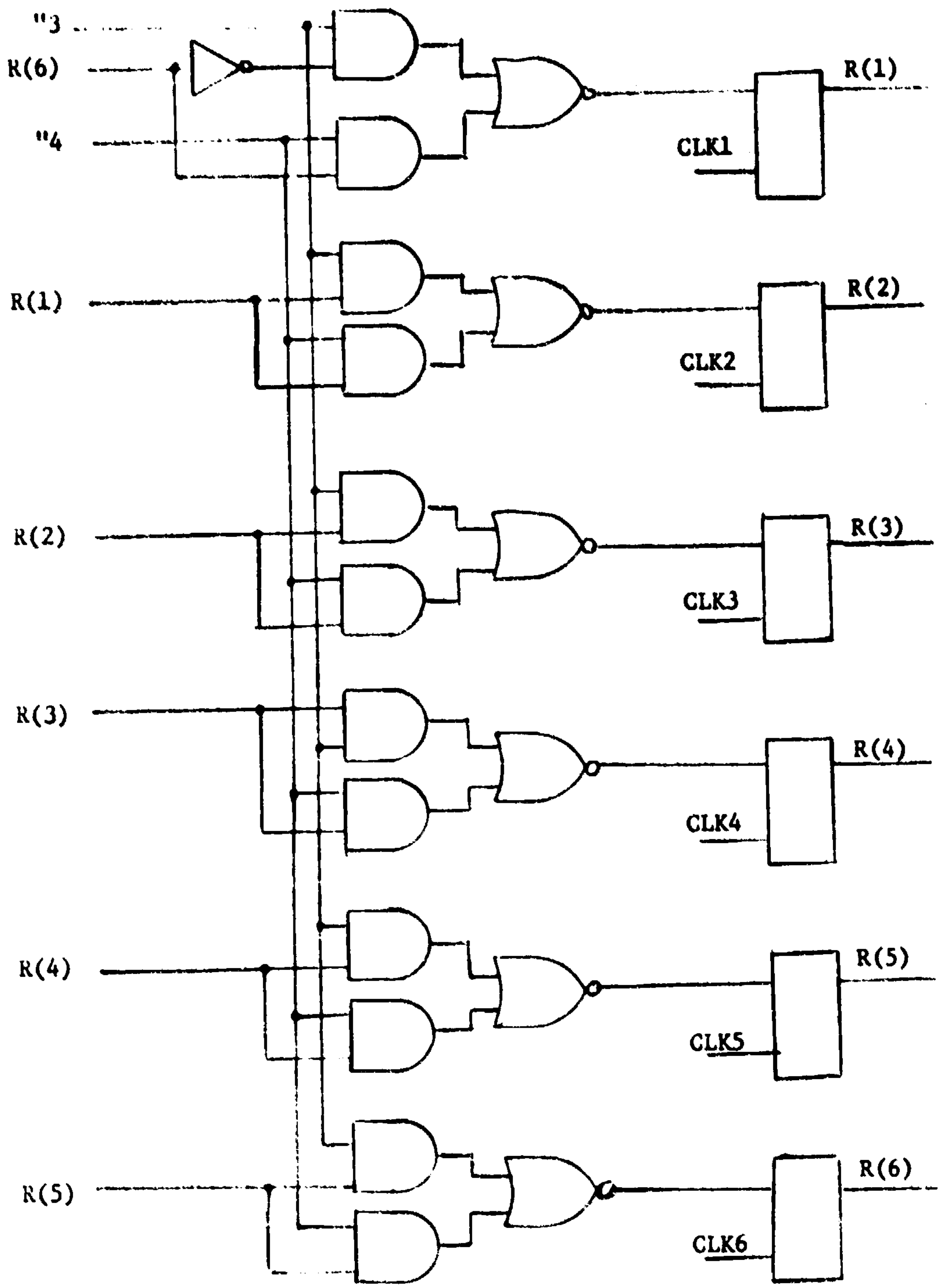


Figure 19(f): (Continued)

is drawn without the state declaration. The register transfer occurs with the clock and other BE output is independent of the state declaration.

6.3 MEMORY

Figure 20 shows a DDL description for 4-words, 4 bits/word memory, along with the DDLTRN output, simulation commands, simulation output, and the circuit using the synthesis output. A detailed description of Figure 20(a) follows:

Line 1: The name of the system is MEMORY

Line 2: Four words of the memory are declared as 4-registers R1, R2, R3, and R4, each 4 bits long (numbered as 1 through 4). Register OU is also 4 bits long (numbered as 1 through 4), used as a buffer.

Line 3: A single phase clock (time) ME.

Line 4: Input terminals RE and WR, 1 bit long, for READ and WRITE operations respectively. Terminal S, 2 bits long, to provide word address.

Line 5: Automaton MEM is controlled by Clock ME.

Lines 6-9: The READ and WRITE operations take place when clock (ME) is present. The register (word) is selected depending on the value of S. The contents of the register (buffer) OU is transferred to the selected register (word) if the WR signal is on. The contents of the selected register (word) is transferred to the register (buffer) OU if the RE signal is on.

Line 10: Sets the flags of DDLTRN to output the results of each of the six phases and the facility table.

Figure 20(b) shows the input commands for DDLSIM. Flags for DDLSIM are set for decimal data input (4) and binary data output (6) in line 1. Registers (words) R1, R2, R3, and R4 are initialized with

```
1:   <SY>MEMORY:
2:   <RE>R1(4),R2(4),R3(4),R4(4),DD(4).
3:   <I1>MA.
4:   <IE>RE,WR,S(2).
5:   <AU>MEM:MA:
6:       IS #00(2)RE]DD<=R1,,]WR]R1<=DD.
7:                      #01(2)RE]DD<=R2,,]WR]R2<=DD.
8:                      #02(2)RE]DD<=R3,,]WR]R3<=DD.
9:                      #03(2)RE]DD<=R4,,]WR]R4<=DD....
10:  <FL>3,4,5,6,8.
```

Figure 20(a): Description for Memory

```
PASS1--FACILITIES IDENTIFIED



DECLARED FACILITIES

      <SY> MEMORY
      <RE> R1(1:4)
           R2(1:4)
           R3(1:4)
           R4(1:4)
           DD(1:4)
      <I1> MA(1:1)
      <IE> RE(1:1)
           WR(1:1)
           S(1:2)
           <AU> MEM
```

Figure 20(a): (Continued)

```
PASS6--SUBFACILITIES DISJOINED

<SY> NETFRY: "1=(*/S'0U2   ),
     "2="1*HF,
     "3="1*AH,
     "4=(*/S'1U2   ),
     "5="4*HF,
     "6="4*AH,
     "7=(*/S'2U2   ),
     "8="7*HF,
     "9="7*AH,
     "10=(*/S'3U2   ),
     "11="10*HF,
     "12="10*AH,
     ]NE*"2 + NE*"5 + NE*"8 + NE*"11] CC<-"2*H1 + "5*H2 + "8*H3 + "11*H4.,
     ]NE*"3] H1<-"3*CC.,
     ]NE*"6] H2<-"6*CC.,
     ]NE*"9] H3<-"9*CC.,
     ]NE*"12] H4<-"12*CC.,
     .
```

Figure 21(c): (Continued)

```
DIGITAL DESIGN LANGUAGE SIMULATOR      VERSION MSFC 1979

        1:      <FL>4,6
        2:      <IN>R1,R2,R3,R4/1,2,3,4
        3:      <IN>OU/4
        4:      <TR>TR/TME/
        5:      <RE>TR/RF,RR,S/0,1,0,1,0,1,0,1,2,1,0,3
        6:      <OU>TR/R1,R2,R3,R4,RE,RR,OU,S/
        7:      <SI>
```

Figure 20(b): Simulation Commands

```
DIGITAL DESIGN LANGUAGE SIMULATOR      VERSION MSFC 1979
```

TIME	R1	R2	R3	R4	RE	RR	OU	S
0	0001	0010	0011	0100	0	0	0100	00
2	0100	0010	0011	0100	0	1	0100	00
4	0100	0010	0011	0100	1	0	0010	01
6	0100	0010	0010	0100	0	1	0010	10
8	0100	0010	0010	0100	1	0	0100	11

```
END      OF FILE REACHED ON INPUT
SIMULATION TERMINATED AT TIME =          [illegible]
```

Figure 20(c): Simulation Output

IDENTIFIER TABLE

NO.	IDENTIFIER	NO.	IDENTIFIER	NO.	IDENTIFIER	NO.	IDENTIFIER
1	*1(1)	2	S(1)	3	S(2)	4	XXXXXXXX(1)
5	XXXXXXXX(1)	6	*2(1)	7	RE(1)	8	*3(1)
9	AR(1)	10	*4(1)	11	*5(1)	12	*6(1)
13	*7(1)	14	*8(1)	15	*9(1)	16	*10(1)
17	*11(1)	18	*12(1)	19	RE(1)	20	OU(1)
21	R1(1)	22	R2(1)	23	R3(1)	24	R4(1)
25	XXXXXXXX(1)	26	XXXXXXXX(1)	27	XXXXXXXX(1)	28	XXXXXXXX(1)
29	OU(2)	30	R1(2)	31	R2(2)	32	R3(2)
33	R4(2)	34	XXXXXXXX(1)	35	XXXXXXXX(1)	36	OU(3)
37	R1(3)	38	R2(3)	39	R3(3)	40	R4(3)
41	XXXXXXXX(1)	42	XXXXXXXX(1)	43	OU(4)	44	R1(4)
45	R2(4)	46	R3(4)	47	R4(4)	48	XXXXXXXX(1)
49	XXXXXXXX(1)	50	XXXXXXXX(1)	51	XXXXXXXX(1)	52	XXXXXXXX(1)
53	XXXXXXXX(1)	54	XXXXXXXX(1)	55	XXXXXXXX(1)	56	XXXXXXXX(1)
57	XXXXXXXX(1)	58	XXXXXXXX(1)	59	XXXXXXXX(1)	60	XXXXXXXX(1)
61	XXXXXXXX(1)	62	XXXXXXXX(1)	63	XXXXXXXX(1)	64	XXXXXXXX(1)
65	XXXXXXXX(1)	66	XXXXXXXX(1)	67	XXXXXXXX(1)	68	XXXXXXXX(1)
69	XXXXXXXX(1)						

Figure 20(d): Synthesis Output

CELL TABLE

CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL
1	1310	2	1310	3	1620	4	1620	5	1620	6	1620	7	1620
8	1620	9	1620	10	1620	11	1620	12	1620	13	1620	14	1620
15	1800	16	1310	17	1800	18	1310	19	1830	20	1800	21	1310
22	1830	23	1800	24	1310	25	1830	26	1800	27	1310	28	1830
29	1620	30	1620	31	1830	32	1620	33	1830	34	1620	35	1830
36	1620	37	1830	38	1620	39	1620	40	1830	41	1620	42	1830
43	1620	44	1830	45	1620	46	1830	47	1620	48	1620	49	1830
50	1620	51	1830	52	1620	53	1830	54	1620	55	1830	56	1620
57	1620	58	1830	59	1620	60	1830	61	1620	62	1830	63	1620
64	1830												

Figure 20(d): (Continued)

NET TABLE

NET	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	P
1	3	4	4	3	5	3														
2	1	2	9	3	12	3														
3	2	2	6	2	12	2														
4	1	3	3	3	6	3														
5	2	3	3	2	9	2														
6	6	4	15	8	17	9	20	9	23	9	26	9								
7	4	2	7	2	10	2	13	2												
8	5	4	29	2	30	3	32	3	34	3	36	3								
9	5	2	6	2	11	2	14	2												
10	8	4	7	3	8	3														
11	7	4	15	6	17	7	20	7	23	7	26	7								
12	8	4	38	2	39	3	41	3	43	3	45	3								
13	9	4	10	3	11	3														
14	10	4	15	4	17	5	20	5	23	5	26	5								
15	11	4	47	2	48	3	50	3	52	3	54	3								
16	12	4	13	3	14	3														
17	13	4	15	2	17	3	20	3	23	3	26	3								
18	14	4	56	2	57	3	59	3	61	3	63	3								
19	15	3	15	5	15	7	15	9	29	3	38	3	47	3	56	3				
20	16	4	30	2	39	2	48	2	57	2										
21	17	8	31	4																
22	17	6	40	4																
23	17	4	49	4																
24	17	2	58	4																
25	15	10	16	2																
26	18	3	19	2	22	2	25	2	28	2										
27	17	10	18	2																
28	18	3	19	3																
29	22	4	32	2	41	2	50	2	59	2										
30	20	8	33	4																
31	20	6	42	4																
32	20	4	51	4																
33	20	2	60	4																
34	20	10	21	2																
35	21	3	22	3																
36	25	4	34	2	43	2	52	2	61	2										
37	23	8	35	4																
38	23	6	44	4																
39	23	4	53	4																
40	23	2	62	4																
41	23	10	24	2																
42	24	3	25	3																
43	28	4	36	2	45	2	54	2	63	2										
44	26	8	37	4																
45	26	6	46	4																
46	26	4	55	4																
47	26	2	64	4																
48	26	10	27	2																
49	27	3	28	3																
50	29	4	31	2	33	2	35	2	37	2										
51	30	4	31	3																
52	32	4	33	3																
53	34	4	35	3																

Figure 20(d): (Continued)

54	36	4	37	3						
55	38	4	40	2	42	2	44	2	46	2
56	39	4	40	3						
57	41	4	42	3						
58	43	4	44	3						
59	45	4	46	3						
60	47	4	49	2	51	2	53	2	55	2
61	48	4	49	3						
62	50	4	51	3						
63	52	4	53	3						
64	54	4	55	3						
65	56	4	58	2	60	2	62	2	64	2
66	57	4	58	3						
67	59	4	60	3						
68	61	4	62	3						
69	63	4	64	3						

Figure 20(d): (Continued)

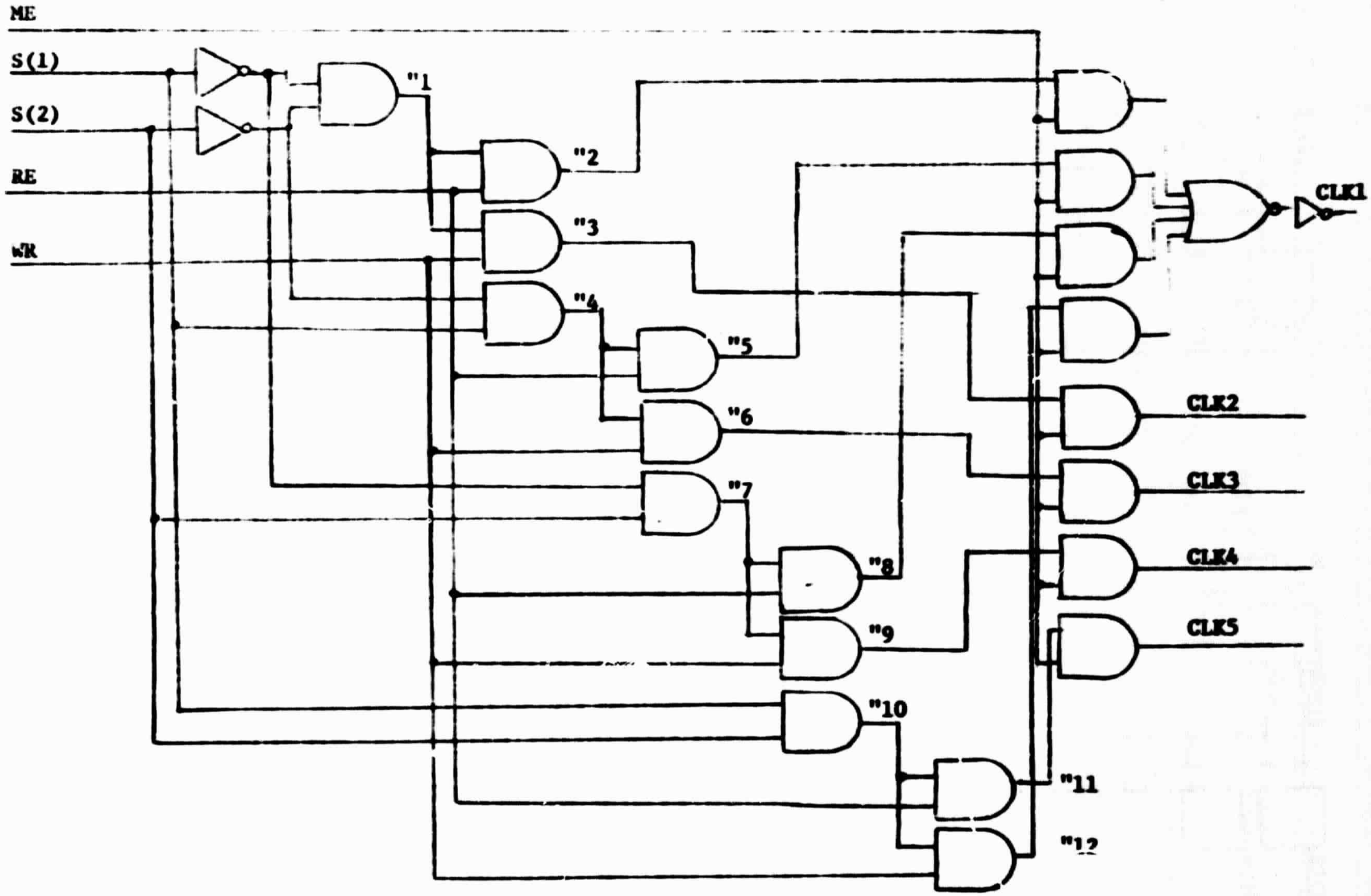


Figure 20(e): Memory Circuit

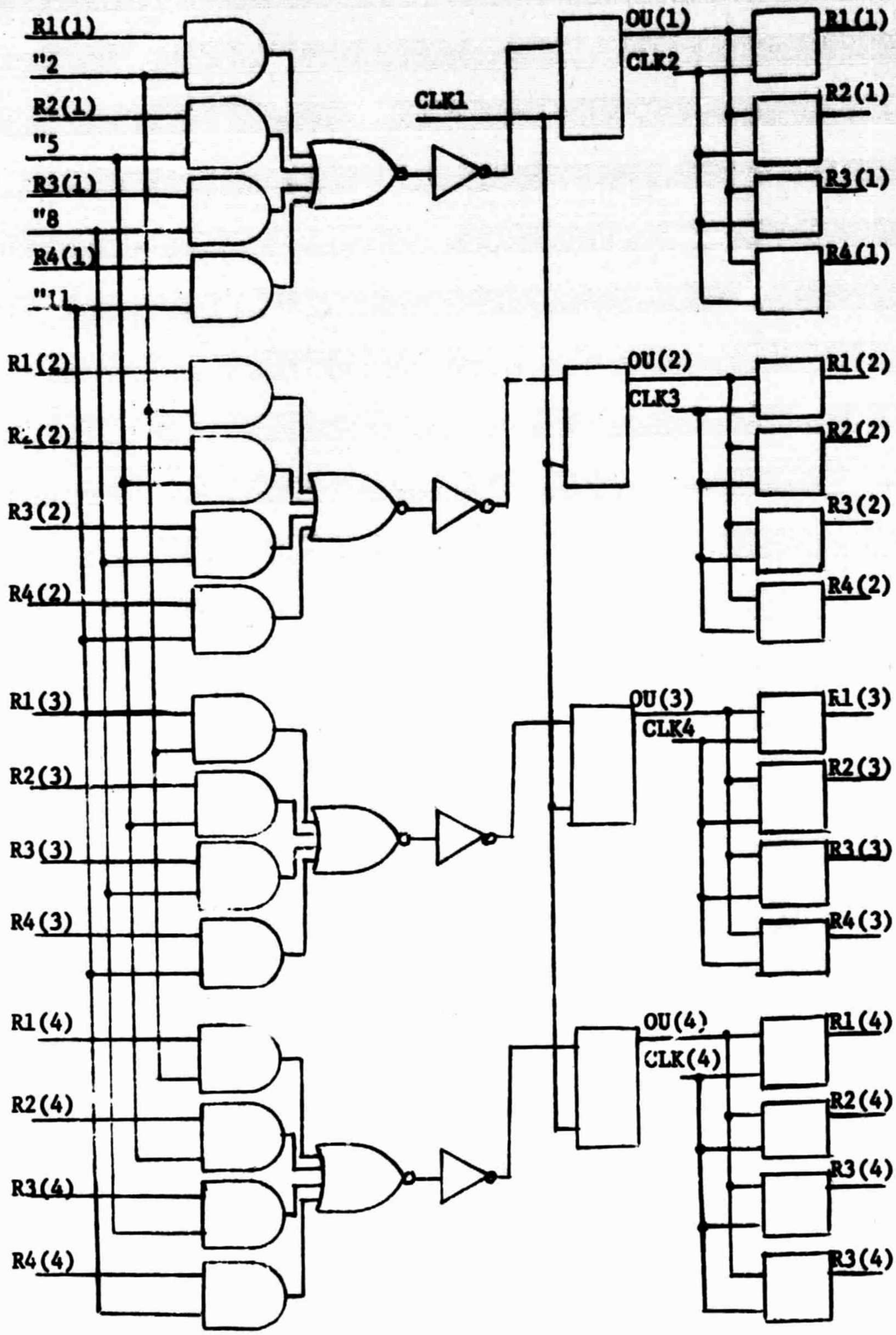


Figure 20(e): (Continued)

1, 2, 3, and 4 respectively in line 2. The register OU is initialized to 4 in line 3. An input/output trigger TR (raising edge of ME) is declared in line 4. Line 5 reads in the values for RE, WR, and S when the TR is on. (There are 4 sets of values). The values of RI, R2, R3, R4, RE, WR, OU, and S are to be output when TR is on (line 6). The simulation is started with the <SI> in line 7. Figure 20(c) is the simulation output. Figure 20(d) shows the synthesis output for the memory circuit.

Figure 20(e) shows a circuit diagram for the 4 words, 4 bit/word memory circuit drawn from the synthesis output. The automatic design from the DDL description output does not add any more cells compared to the manual design.

6.4 VARIABLE TIMER

Figure 21 shows a variable timer circuit, along with the DDL description, DDLTRN output, simulation commands, simulation output, synthesis output, and the circuit using one synthesis output. Figure 21(a) is the block diagram for the variable timer circuit. The circuit consists of a divide by 3600 circuit R along with a counter CT that counts the number of times R goes to zero. CT is compared with the input setting IN. If IN equals CT, an output pulse is given, and the start register is reset to disable the clock. ABORT input clears R, CT and START. START input sets START. A detailed description of Figure 21(c) follows:

<u>Line 1</u>: The name of the system is VTIMR.

<u>Line 2</u>: Declares 12 bits register R, 6 bits registers CT andIN (numbered 1 through 21). START and ABORT are 1 bit registers.

<u>Line 3</u>: A single phase clock (time) P.

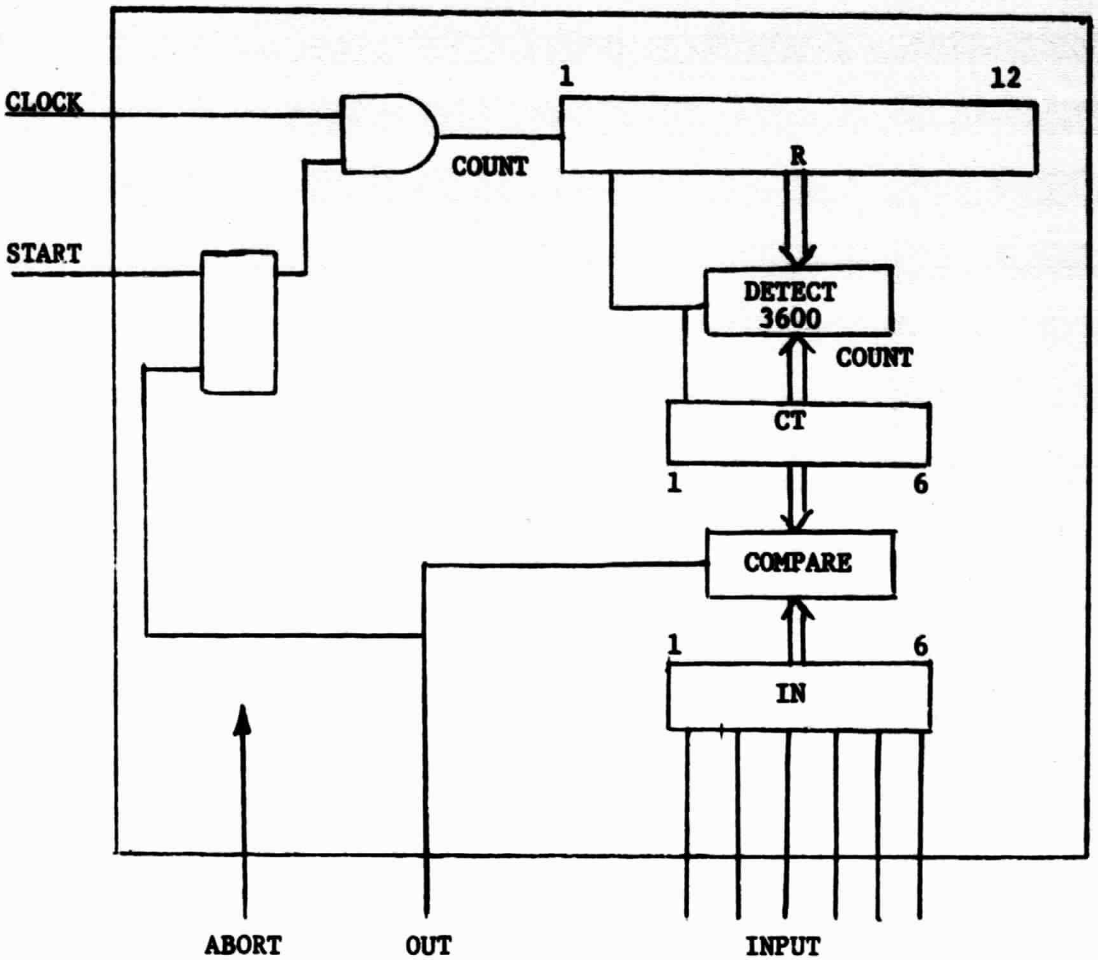


Figure 21(a): Variable Timer

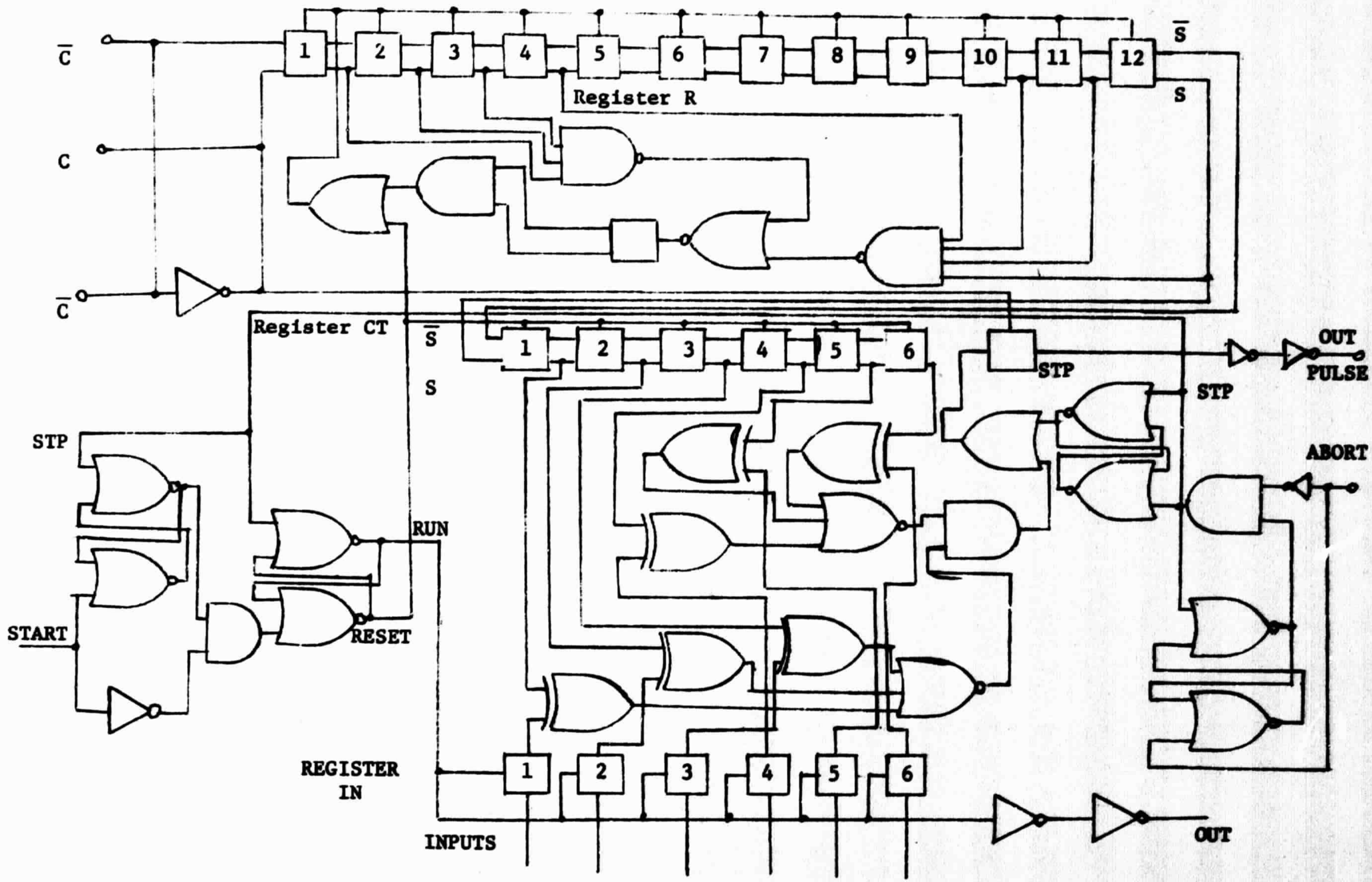


Figure 21(b): Variable Times Circuit

```
<SY>VTIMER:
<RE>R(12),CT(6),IN(6),START,ABORT.
<TI>P.
<TE>OUT,X,Y,XA(6).
<RO>XA=CT*IN.
<RO>Y=R(1)*R(2)*R(3)*R(9)*R(10)*R(11)*R(12),X=↑(+/XA).
<OP>CNTUP(12)$Z$
  <TE>Z(12),C(12).
  <RO>CC=(C(2:12)||1).
  <RO>C=Z*CC,CNTUP=Z⊕CC..
<AU>VTMR(2):P:
  <ST>T(0):START:)ABORT))CT<-0.,ABORT<-0,OUT=0,->S.
  S(1):R<-CNTUP$R$,)Y)->T;->S..
  T(2):CT<-CNTUP(7:12)$0U6(CT$,)X)OUT=1,ABORT<-1.,->T....
<FL>3,4,5,6,8.
```

Figure 21(c): Description for Variable Timer

```
PASSE--SUBFACILITIES DISJOINED

<SY> VTIPER: I=*/VIRR'002,
     S=*/VIRR'102,
     T=*/VIRR'202,
     "1=I*START,
     "2="1*ABORT,
     "3=S*Y, "4=S*TY,
     "5=I*X, XA=(CI∧IR),
     Y=R(1)*R(2)*R(3)*R(9)*R(10)*R(11)*R(12),
     X=T(+/XA),
       C(1:11)=Z(1:11)*C(2:12),
     C(12)=Z(12)*1D1   ,
       CNTUP(1:11)=(Z(1:11)∧C(2:12)),
     CNTUP(12)=(Z(12)∧1D1   ),
     ]P*"2 + P*T] CT<-"2*0 + I*CNTUP(7:12).,
     ]P*"1 + P*"5] ABORT<-"1*0 + "5*1D1   .,
     OUT="1*0 + "5*1D1   ,
     ]P*"1 + P*"3 + P*"4 + P*T] VIRR<-"1*1D2 + "3*2D2 + "4*1D2 + I*0D2.,
     ]P*S] R<-S*CNTUP.,
        Z(1:6)=S*R(1:6) + I*0D6   ,
     Z(7:12)=S*R(7:12) + T*CT,
     .
```

Figure 21(a): (Continued)

```
1:     <FL>4,8
2:     <IN>IN/5
3:     <IN>ABORT,START/1,1
4:     <LO>I/R/3595
5:     <TR>TR/TP/
6:     <OU>TR/R,CT,IN,ABORT,START,CUT,VTPR/
7:     <ST>120
8:     <ST>
```

Figure 21(d): Simulation Commands

TIME	R	CT	IN	ABORT	START	OUT	VTPR
0	0000	00	05	1	1	0	0
2	3595	00	05	0	1	0	1
4	3596	00	05	0	1	0	1
6	3597	00	05	0	1	0	1
8	3598	00	05	0	1	0	1
10	3599	00	05	0	1	0	1
12	3600	00	05	0	1	0	2
14	3600	01	05	0	1	0	0
16	3595	01	05	0	1	0	1
18	3596	01	05	0	1	0	1
20	3597	01	05	0	1	0	1
22	3598	01	05	0	1	0	1
24	3599	01	05	0	1	0	1
26	3600	01	05	0	1	0	2
28	3600	02	05	0	1	0	0
30	3595	02	05	0	1	0	1
32	3596	02	05	0	1	0	1
34	3597	02	05	0	1	0	1
36	3598	02	05	0	1	0	1
38	3599	02	05	0	1	0	1
40	3600	02	05	0	1	0	2
42	3600	03	05	0	1	0	0
44	3595	03	05	0	1	0	1
46	3596	03	05	0	1	0	1
48	3597	03	05	0	1	0	1
50	3598	03	05	0	1	0	1
52	3599	03	05	0	1	0	1
54	3600	03	05	0	1	0	2
56	3600	04	05	0	1	0	0
58	3595	04	05	0	1	0	1
60	3596	04	05	0	1	0	1
62	3597	04	05	0	1	0	1
64	3598	04	05	0	1	0	1
66	3599	04	05	0	1	0	1
68	3600	04	05	0	1	0	2
70	3600	05	05	0	1	0	0
72	3595	05	05	0	1	0	1
74	3596	05	05	0	1	0	1
76	3597	05	05	0	1	0	1
78	3598	05	05	0	1	0	1
80	3599	05	05	0	1	0	1
82	3600	05	05	0	1	1	2
84	3600	06	05	1	1	0	0
86	3595	00	05	0	1	0	1
88	3596	00	05	0	1	0	1
90	3597	00	05	0	1	0	1
92	3598	00	05	0	1	0	1
94	3599	00	05	0	1	0	1
96	3600	00	05	0	1	0	2
98	3600	01	05	0	1	0	0
100	3595	01	05	0	1	0	1
102	3596	01	05	0	1	0	1
104	3597	01	05	0	1	0	1
106	3598	01	05	0	1	0	1
108	3599	01	05	0	1	0	1
110	3600	01	05	0	1	0	2
112	3600	02	05	0	1	0	0
114	3595	02	05	0	1	0	1
116	3596	02	05	0	1	0	1
118	3597	02	05	0	1	0	1

SIMULATION TERMINATED BY A STOP-COMMAND AT TIME= 120

Figure 21(e): Simulation Output

Line 4: Terminals OUT, X, Y are 1 bit long. Terminal XA is 6 bits long.

Lines 5-6: Line 5 compares each bit of CT and IN. If all bits are equal, then in line 6, X is set. Y checks for count in R to be 3600.

Lines 7-10: A special operator named by CNTUP. The output of the operator is 12 bits number. The input is through the argument Z (Z is a formal parameter). The operator CNTUP is simply an add 1 circuit.

Line 11: Automaton VTMR is controlled by the clock P. VTMR is 2 bits long, thus can have a maximum of 4 states.

Line 12: State I with identification 0. Automaton VTMR waits in I till START is 1. When START is 1, CT is reset if ABORT is 1; ABORT and OUT are reset, and a transition to state S is made (all in parallel). The period terminates I.

Line 13: State S with the designation 1. R is incremented by 1 (R is counted up from 3595 rather than zero, to save some simulation steps). Transition is made to state T, if Y is set, else remains in the same state. The periods terminate IF-THEN and state S.

Line 14: State T with the designation 2. CT is incremented by 1. OUT and ABORT are reset if X is set. Unconditional transition to state I is made. The periods at the end of line 14 terminate IF-THEN, state T, ST, AU and SY declarations respectively.

Line 15: Sets the flags of DDLTRN to output results at each of the six phases and facility table.

Figure 21(d) shows the input commands for DDLSIM. Flags for DDLSIM are set for decimal data input (4) and decimal data output (8) in line 1. Line 2 initializes IN with value 5. Line 3 initializes ABORT and START with 1. Line 4 loads R with 3595 whenever in I state.

IDENTIFIER TABLE

NO.	IDENTIFIER	NO.	IDENTIFIER	NO.	IDENTIFIER	NO.	IDENTIFIER
1	I(1)	2	VIMH(1)	3	VIMH(2)	4	XXXXXXXX(1)
5	XXXXXXXX(1)	6	S(1)	7	I(1)	8	*1(1)
9	START(1)	10	*2(1)	11	APORT(1)	12	*3(1)
13	Y(1)	14	*4(1)	15	XXXXXXXX(1)	16	*5(1)
17	X(1)	18	XA(1)	19	CI(1)	20	IN(1)
21	XA(2)	22	CI(2)	23	IN(2)	24	XA(3)
25	CI(3)	26	IN(3)	27	XA(4)	28	CI(4)
29	IN(4)	30	XA(5)	31	CI(5)	32	IN(5)
33	XA(6)	34	CI(6)	35	IN(6)	36	R(1)
37	R(2)	38	R(3)	39	R(9)	40	R(10)
41	R(11)	42	R(12)	43	XXXXXXXX(1)	44	XXXXXXXX(1)
45	XXXXXXXX(1)	46	C(1)	47	Z(1)	48	C(2)
49	Z(2)	50	C(3)	51	Z(3)	52	C(4)
53	Z(4)	54	C(5)	55	Z(5)	56	C(6)
57	Z(6)	58	C(7)	59	Z(7)	60	C(8)
61	Z(8)	62	C(9)	63	Z(9)	64	C(10)
65	Z(10)	66	C(11)	67	Z(11)	68	C(12)
69	Z(12)	70	CNTUP(1)	71	CNTUP(2)	72	CNTUP(3)
73	CNTUP(4)	74	CNTUP(5)	75	CNTUP(6)	76	CNTUP(7)
77	CNTUP(8)	78	CNTUP(9)	79	CNTUP(10)	80	CNTUP(11)
81	CNTUP(12)	82	P(1)	83	XXXXXXXX(1)	84	XXXXXXXX(1)
85	XXXXXXXX(1)	86	XXXXXXXX(1)	87	XXXXXXXX(1)	88	XXXXXXXX(1)
89	XXXXXXXX(1)	90	XXXXXXXX(1)	91	XXXXXXXX(1)	92	XXXXXXXX(1)
93	OUT(1)	94	XXXXXXXX(1)	95	XXXXXXXX(1)	96	XXXXXXXX(1)
97	XXXXXXXX(1)	98	XXXXXXXX(1)	99	XXXXXXXX(1)	100	XXXXXXXX(1)
101	XXXXXXXX(1)	102	R(4)	103	XXXXXXXX(1)	104	R(5)
105	XXXXXXXX(1)	106	R(6)	107	XXXXXXXX(1)	108	XXXXXXXX(1)
109	R(7)	110	XXXXXXXX(1)	111	R(8)	112	XXXXXXXX(1)
113	XXXXXXXX(1)	114	XXXXXXXX(1)	115	XXXXXXXX(1)	116	XXXXXXXX(1)
117	XXXXXXXX(1)	118	XXXXXXXX(1)	119	XXXXXXXX(1)	120	XXXXXXXX(1)
121	XXXXXXXX(1)	122	XXXXXXXX(1)	123	XXXXXXXX(1)	124	XXXXXXXX(1)
125	XXXXXXXX(1)						

Figure 21(f): Synthesis Output

CELL TABLE

CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL
1	1310	2	1310	3	1620	4	1620	5	1620	6	1620	7	1620
8	1620	9	1310	10	1620	11	1620	12	2310	13	2310	14	2310
15	2310	16	2310	17	2310	18	1640	19	1640	20	1740	21	1720
22	1120	23	1620	24	1620	25	1620	26	1620	27	1620	28	1620
29	1620	30	1620	31	1620	32	1620	33	1620	34	2310	35	2310
36	2310	37	2310	38	2310	39	2310	40	2310	41	2310	42	2310
43	2310	44	2310	45	1310	46	1870	47	1310	48	1620	49	1830
50	1620	51	1830	52	1620	53	1830	54	1620	55	1830	56	1620
57	1830	58	1620	59	1830	60	1870	61	1310	62	1830	63	1800
64	1300	65	1310	66	1720	67	1830	68	1830	69	1620	70	1620
71	1830	72	1620	73	1830	74	1620	75	1830	76	1620	77	1830
78	1620	79	1830	80	1620	81	1300	82	1830	83	1620	84	1830
85	1620	86	1830	87	1620	88	1830	89	1300	90	1620	91	1830
92	1620	93	1830	94	1620	95	1830	96	1620	97	1620	98	1300
99	1620	100	1620	101	1620	102	1620	103	1870	104	1310	105	1870
106	1310	107	1870	108	1310	109	1870	110	1300	111	1310	112	1870
113	1310	114	1870	115	1310								

Figure 21(f): (Continued)

NET TABLE

NET	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	PIN
1	3	4	6	3																		
2	1	2	5	3	67	4																
3	2	2	4	2	68	4																
4	1	3	3	3	4	3																
5	2	3	3	2	5	2																
6	4	4	8	3	10	3	69	2	70	3	72	3	74	3	76	3	78	3	81	2		
7	5	4	11	3	46	2	48	3	50	3	52	3	54	3	56	3	58	3	64	2		
8																						
9	8	2																				
10	7	4	46	4																		
11	7	2	62	4																		
12	8	4	64	6																		
13	8	2	9	2	19	6																
14	10	4	64	4	66	2																
15	9	3	10	2																		
16	11	4	60	2																		
17	11	2	22	4																		
18	12	4	20	5																		
19	12	2	49	4	103	2	105	2	107	2	109	2	112	2	114	2						
20	12	3																				
21	13	4	20	4																		
22	13	2	51	4																		
23	13	3																				
24	14	4	20	3																		
25	14	2	53	4																		
26	14	3																				
27	15	4	20	2																		
28	15	2	55	4																		
29	15	3																				
30	16	4	21	3																		
31	16	2	57	4																		
32	16	3																				
33	17	4	21	2																		
34	17	2	59	4																		
35	17	3																				
36	19	5	71	4	96	2																
37	19	4	73	4	97	2																
38	19	3	75	4	99	2																
39	18	5	89	4	107	4																
40	18	4	91	4	109	4																
41	18	3	93	4	112	4																
42	18	2	95	4	114	4																
43	18	6	19	2																		
44	20	6	22	2																		
45	21	4	22	3																		
46	23	4																				
47	23	3	34	2	96	4																
48	23	2	24	4	34	3																
49	24	3	35	2	98	4																
50	24	2	25	4	35	3																
51	25	3	36	2	99	4																
52	25	2	26	4	36	3																
53	26	3	37	2	100	4																

Figure 21(f): (Continued)

54	26	2	27	4	37	3												
55	27	3	38	2	101	4												
56	27	2	28	4	38	3												
57	28	3	39	2	102	4												
58	28	2	29	4	39	3												
59	29	3	40	2	104	3												
60	29	2	30	4	40	3												
61	30	3	41	2	106	3												
62	30	2	31	4	41	3												
63	31	3	42	2	108	3												
64	31	2	32	4	42	3												
65	32	3	43	2	111	3												
66	32	2	33	4	43	3												
67	33	3	44	2	113	3												
68	33	2	44	3	45	2												
69																		
70	34	4	48	2	70	2												
71	35	4	50	2	72	2												
72	36	4	52	2	74	2												
73	37	4	54	2	76	2												
74	38	4	56	2	78	2												
75	39	4	58	2	80	2												
76	40	4	83	2														
77	41	4	85	2														
78	42	4	87	2														
79	43	4	90	2														
80	44	4	92	2														
81	45	3	94	2														
82	46	3	46	5	80	3	60	5	64	3	64	5	64	7	64	9	69	3
83	48	6	47	2														
84	47	3	49	2	51	2	53	2	55	2	57	2	59	2				
85	48	4	49	3														
86	50	4	51	3														
87	52	4	53	3														
88	54	4	55	3														
89	56		57	3														
90	58	4	59	3														
91	60	8	61	2														
92	61	3	62	2														
93	8	4																
94	64	3	64	2	103	3	105	3	107	3	109	3	112	3	114	3		
95	64	10	65	2														
96	65	3	67	2	68	2												
97	66	4	67	3														
98	69	4	71	2	73	2	75	2	77	2	79	2	82	2	84	2	86	2	89	2
99	70	4	71	3														
100	72	4	73	3														
101	74	4	75	3														
102	77	4	100	2														
103	76	4	77	3														
104	79	4	101	2														
105	78	4	79	3														
106	82	4	102	2														
107	81	3	81	3	83	3	85	3	87	3	90	3	92	3	94	3	96	3	98	2
108	81	4	82	3														
109	84	4	103	4														
110	83	4	84	3														
111	88	4	105	4														
112	85	4	88	3														

Figure 21(f): (Continued)

113	87	4	89	3																				
114	89	3	89	2	91	2	93	2	95	2														
115	90	4	91	3																				
116	92	4	93	3																				
117	94	4	95	3																				
118	98	3	98	3	99	3	100	3	101	3	102	3	103	5	105	5	107	5	110	2				
119	103	8	104	2																				
120	105	8	106	2																				
121	107	8	108	2																				
122	110	3	110	5	112	5	114	5																
123	110	6	111	2																				
124	112	8	113	2																				
125	114	8	115	2																				

Figure 21(f): (Continued)

Line 5 is output trigger TR (raising edge of P). Line 6 outputs R, CT, IN, ABORT, START, OUT, VTMR when TR is on. Line 7 stops simulation after time 120. The simulation is started with <SI> in line 8. The simulation output is given in Figure 21(e). Figure 21(f) gives synthesis output for the variable timer circuit.

Figure 21(b) shows the manual circuit for the variable timer. There are more cells in an automatic design compared to manual design. The excessive cells are due to the following:

(1) The manual design uses a binary counter. The counter in DDL description is described by add 1 circuit. This adds 24 more cells for line 10 of Figure 21(c).

(2) The use of the states in an automaton adds extra cells that are required in the automatic design. (The first 9 BEs which describe states, add 8, 2-input AND cells and 2-inverters. The condition portion of RTEs add 7 cells (1 standard cell 1800, 2-1870s, 1-1620, and 3-inverters, along with the 2-1930s to declare VTMR register).

6.5 MINICOMPUTER

Figure 22 shows a minicomputer description along with the DDLTRN output, simulation commands, simulation output, and synthesis output. For details of the DDL description Figure 22(a), refer to [4].

Figure 22(b) shows the input commands for DDLSIM. Flags for DDLSIM are set for decimal data input and output (4 and 8) in line 1. The memory locations 0 to 14 are initialized in lines 2 to 4. Line 5 reads PC value whenever in IN state. Line 6 initializes START with one. Line 7 outputs CPU, IR, PC, MAR whenever in IN state. Lines 8,9 output MAR, MBR, IR, ACC whenever in FE, and EX states. The simulation is started with line 10. Figure 22(c) shows the simulation output.

```
<SY>MINI:
<RE>MAR(0:7),MBR(0:11),PC(0:7),ACC(0:11).
<RE>IR(0:11)=OP(3)!IBIT!ADR(8),RUN.
<RE> X(0:11).
<ME>M(256:12).
<TE>MBUS(12).
<LA>START.
<TE>P(4).
<TI>R.
<OP>CNTUP(8)$X$
    <TE>X(8),C(8).
    <TD>CC=(C(2:8)!1B1).
    <BO>C=X*CC,CNTUP=X⊕CC..
<OP>SUM(12)$X,Y$
    <TE>X(12),C(12),Y(12),COUT(12).
    <TD>CIN=COUT(2:12)!0B1.
```

Figure 22(a): Description for Minicomputer

```
<BU>COUT=X*Y+X*CIN+Y*CIN,
        SUM=X@Y@CIN..
<AU>CLK(2):R:
    <ST>S(0):START:P=8D4,->I.
        I(1):P=4D4,->J.
        J(2):P=2D4,->L.
        L(3):P=1D4,->S...
<AU>CPU(4):R:
    <ST>IN(0):START:IF(4)]ACC<-0,MAR<-PC,MBR<-0,X<-0,
                    RUN<-1,->FE..
        FE(1):RUN:IP(1)]MAR<-PC..IP(2)]PC<-CNTUP$PC$,
                  MBUS=M(MAR),MBR<-MBUS..IP(3)]IR<-MBR..
                  IP(4)]]OP(1)*OP(2)*OP(3)]RUN<-0,->IN;
                  ]IR11]->DEF;->EX....
        DEF(2):IP(1)]MAR<-ADR..IP(2)]MBUS=M(MAR),MBR<-MBUS..
               IP(3)]ADR<-MBR(4:11)..IP(4)]->EX..
        EX(3):IP(4)::OPA0D3->XAND #1D3 ->XAND #2D3 ->XISZ
              #3D3 ->XDCA #4D3 ->XJSR #5D3 ->XJMP #6D3 ->XRET..
```

Figure 22(a): (Continued)

```
XAND(4):]P(1)]X<-ACC.,]P(2)]MAR<-ADR.,
        ]P(3)]MBUS=M(MAR),MBR<-MBUS.,
        ]P(4)]]TOP(3)]ACC<-MBR*X;ACC<-SUM$MBR,X$.,->FE..
XISZ(5):]P(1)]MAR<-ADR.,
        ]P(2)]MBUS=M(MAR),MBR<-MBUS.,
        ]P(3)]MBR<-SUM$MBR,1D12$.,
        ]P(4)]MBUS=MBR,M(MAR)<-MBUS,]↑(+/MBR)]
              PC<-CNTUP$PC$;,->FE..
XDCA(6):]P(1)]MBR<-ACC.,]P(2)]MAR<-ADR.,
        ]P(3)]ACC<-0,MBUS=MBR,M(MAR)<-MBUS.,]P(4)]->FE..
XJSR(7):]P(1)]MBR<-OD4(PC.,]P(2)]MAR<-0.,
        ]P(3)]MBUS=MBR,M(MAR)<-MBUS.,
        ]P(4)]PC<-ADR,->FE..
 XRET(8):]P(1)]MAR<-0.,]P(2)]MBUS=M(MAR),
          MBR<-MBUS.,]P(4)]PC<-MBR(4:11),->FE..
  XJMP(9):]P(1)]PC<-ADR.,]P(4)]->FE.....
<FL>↑1,↑2.
```

Figure 22(a): (Continued)

PASS6--SUBFACILITIES DISJOINED

```
<SY> MINI: S=*/CLK'002,
     T=*/CLK'102,
     J=*/CLK'202,
     L=*/CLK'302,
     IN=*/CPU'004   ,
     FE=*/CPU'104   ,
     DEF=*/CPU'204   ,
     EX=*/CPU'304,
     XAND=*/CPU'404   ,
     XISZ=*/CPU'504,
     XDCA=*/CPU'604,
     XJSR=*/CPU'704,
     XRET=*/CPU'804   ,
     XJMP=*/CPU'904,
     "1=S*START,
     "2=IN*START,
     "3="2*P(4),
     "4=FE*RUN,
     "5="4*P(1),
     "6="4*P(2),
     "7="4*P(3),
     "8="4*P(4),
     "9="8*IR(0)*IR(1)*IR(2),
     "10="8*↑(IR(0)*IR(1)*IR(2)),
     "11="10*IR(3),
     "12="10*↑IR(3),
     "13=DEF*P(1),
     "14=DEF*P(2),
     "15=DEF*P(3),
     "16=DEF*P(4),
     "17=EX*P(4),
     "18="17*(*/IR(0:2)'0D3   ),
     "19="17*(*/IR(0:2)'1D3   ),
     "20="17*(*/IR(0:2)'2D3   ),
     "21="17*(*/IR(0:2)'3D3   ),
     "22="17*(*/IR(0:2)'4D3   ),
     "23="17*(*/IR(0:2)'5D3   ),
     "24="17*(*/IR(0:2)'6D3   ),
     "25=XAND*P(1),
     "26=XAND*P(2),
     "27=XAND*P(3),
     "28=XAND*P(4),
     "29="28*↑IR(2),
     "30="28*↑(↑IR(2)),
     "31=XISZ*P(1),
     "32=XISZ*P(2),
     "33=XISZ*P(3),
     "34=XISZ*P(4),
     "35="34*↑(*/MBR),
     "36=XDCA*P(1),
     "37=XDCA*P(2),
     "38=XDCA*P(3),
     "39=XDCA*P(4),
     "40=XJSR*P(1),
     "41=XJSR*P(2),
     "42=XJSR*P(3),
     "43=XJSR*P(4),
     "44=XRET*P(1),
     "45=XRET*P(2),
     "46=XRET*P(4),
     "47=XJMP*P(1),
     "48=XJMP*P(4),
```

Figure 22(a): (Continued)

```
  C(1:7)=X"1(1:7)*C(2:8),
C(8)=X"1(8)*1D1   ,
  CNTUP(1:7)=(X"1(1:7)@C(2:8)),
CNTUP(8)=(X"1(8)@1D1   ),
  COUT(1:11)=(X"2(1:11)*Y(1:11) + X"2(1:11)*COUT(2:12) + Y(1:11)*COUT(2:12)),
COUT(12)=(X"2(12)*Y(12) + X"2(12)*0D1    + Y(12)*0D1   ),
  SUM(1:11)=(X"2(1:11)@Y(1:11)@COUT(2:12)),
SUM(12)=(X"2(12)@Y(12)@0D1   ),
F="1*8D4    + I*4D4    + J*2D4    + L*1D4   ,
]R*"1 + R*I + R*J + R*L] CLK<-"1*1D2 + I*2D2 + J*3D2 + L*0D2.,
]R*"3 + R*"29 + R*"30 + R*"38] ACC<-"3*0C1    + "29*MBR*X + "30*SUM + "38*0D1  .,
]R*"3 + R*"5 + R*"13 + R*"26 + R*"31 + R*"37 + R*"41 + R*"44] MAR<-"3*PC + "5*PC + "13*IR(4:11)
      + "26*IR(4:11) + "31*IR(4:11) + "37*IR(4:11) + "41*0D1    + "44*0D1  .,

]R*"3 + R*"25] X<-"3*0D1    + "25*ACC.,
]R*"3 + R*"9] MUN<-"3*1D1    + "9*0D1   .,
]R*"3 + R*"9 + R*"11 + R*"12 + R*"16 + R*"18 + R*"19 + R*"20 + R*"21 + R*"22 + R*"23 + R*"24
      + R*"28 + R*"34 + R*"39 + R*"43 + R*"46 + R*"48] CPU<-"3*1D4    + "9*0D4    + "11*2D4
      + "12*3D4 + "16*3D4 + "18*4D4    + "19*4D4    + "20*5D4 + "21*6D4 + "22*7D4 + "23*9D4 + "24*8D4
      + "28*1D4    + "34*1D4    + "39*1D4    + "43*1D4    + "46*1D4    + "48*1D4  .,
]R*"6 + R*"35 + R*"43 + R*"46 + R*"47] PC<-"6*CNTUP + "35*CNTUP + "43*IR(4:11) + "46*MBR(4:11)
      + "47*IR(4:11).,
X"1="6*PC + "35*PC,
MBUS="6*M(MAR) + "14*M(MAR) + "27*M(MAR) + "32*M(MAR) + "34*MBR + "38*MBR + "42*MBR + "45*M(MAR),

]R*"7] IR(0:3)<-"7*MBR(0:3).,
]R*"7 + R*"15] IR(4:11)<-"7*MBR(4:11) + "15*MBR(4:11).,
X"2="30*MBR + "33*MBR,
Y="30*X + "33*1D12 ,
]R*"34 + R*"38 + R*"42] M(MAR)<-"34*MBUS + "38*MBUS + "42*MBUS.,

]R*"3 + R*"6 + R*"14 + R*"27 + R*"32 + R*"33 + R*"36 + R*"45 + R*"40] MBR(0:3)<-"3*0D4    + "6*MBUS(1:4)
      + "14*MBUS(1:4) + "27*MBUS(1:4) + "32*MBUS(1:4) + "33*SUM(1:4) + "36*ACC(0:3) + "45*MBUS(1:4)
      + "40*0D4  .,
]R*"3 + R*"6 + R*"14 + R*"27 + R*"32 + R*"33 + R*"36 + R*"45 + R*"40] MBR(4:11)<-"3*0D8 + "6*MBUS(5:12)
      + "14*MBUS(5:12) + "27*MBUS(5:12) + "32*MBUS(5:12) + "33*SUM(5:12) + "36*ACC(4:11) + "45*MBUS(5:12)
      + "40*PC.,
.
```

Figure 22(a): (Continued)

```
DIGITAL DESIGN LANGUAGE SIMULATOR      VERSION MSFC 1979       SIMULATION RUN  1

      1:     <FL>4,8
      2:     <IN>M(0:3)/5,6,7,8
      3:     <IN>M(4:7)/4092,0,0,0
      4:     <IN>M(8:14)/5,774,1030,1028,2569,1543,3584
      5:     <RE>IN/PC/8
      6:     <IN>START/1/
      7:     <OU>IN/CPU,IR,PC,MAR/
      8:     <OU>FE/MAR,MBR,IR,PC,ACC/
      9:     <OU>EX/MAR,MBR,IR,PC,ACC/
     10:     <SI>
```

Figure 22(b): Simulation Commands

```
C
P
U  IR    PC MAR MAR   MBR   IR    PC   ACC MAR   MBR   IR    PC   ACC

00 0000 000 000
                     008 0000 0000 008 0000
                                              008 0005 0005 009 0000
                     005 0000 0005 009 0000
                                              006 0000 0768 010 0000
                     000 0005 0768 010 0005
                                              010 1030 1030 011 0005
                     006 0001 1030 011 0005
                                              011 1028 1028 012 0005
                     004 4093 1028 012 0005
                                              012 2569 2569 013 0005
                     012 2569 2569 009 0005
                                              006 0001 0769 010 0005
                     001 0006 0769 010 0011
                                              010 1030 1030 011 0011
                     006 0002 1030 011 0011
                                              011 1028 1028 012 0011
                     004 4094 1028 012 0011
                                              012 2569 2569 013 0011
                     012 2569 2569 009 0011
                                              006 0002 0770 010 0011
                     002 0007 0770 010 0018
                                              010 1030 1030 011 0018
                     006 0003 1030 011 0018
                                              011 1028 1028 012 0018
                     004 4095 1028 012 0018
                                              012 2569 2569 013 0018
                     012 2569 2569 009 0018
                                              006 0003 0771 010 0018
                     003 0008 0771 010 0026
                                              010 1030 1030 011 0026
                     006 0004 1030 011 0026
                                              011 1028 1028 012 0026
                     004 0000 1028 013 0026
                                              013 1543 1543 014 0026
                     007 0026 1543 014 0000
00 3564 015 014

                   END      OF FILE REACHED ON INPUT
                   SIMULATION TERMINATED AT TIME =        457
```

Figure 22(c): Simulation Output

IDENTIFIER TABLE

NO.	IDENTIFIER	NO.	IDENTIFIER	NO.	IDENTIFIER	NO.	IDENTIFIER
1	S(1)	2	CLK(1)	3	CLK(2)	4	XXXXXXXX(1)
5	XXXXXXXX(1)	6	I(1)	7	J(1)	8	L(1)
9	IN(1)	10	CPU(1)	11	CPU(2)	12	CPU(3)
13	CPU(4)	14	XXXXXXXX(1)	15	XXXXXXXX(1)	16	XXXXXXXX(1)
17	XXXXXXXX(1)	18	FF(1)	19	DEF(1)	20	EX(1)
21	XAND(1)	22	XISZ(1)	23	XDCA(1)	24	XJSR(1)
25	XRET(1)	26	XJMP(1)	27	"1(1)	28	START(1)
29	"2(1)	30	"3(1)	31	F(4)	32	"4(1)
33	RUN(1)	34	"5(1)	35	F(1)	36	"6(1)
37	F(2)	38	"7(1)	39	F(3)	40	"8(1)
41	"9(1)	42	IR(0)	43	IR(1)	44	IR(2)
45	"10(1)	46	XXXXXXXX(1)	47	"11(1)	48	IR(3)
49	"12(1)	50	XXXXXXXX(1)	51	"13(1)	52	"14(1)
53	"15(1)	54	"16(1)	55	"17(1)	56	"18(1)
57	XXXXXXXX(1)	58	XXXXXXXX(1)	59	XXXXXXXX(1)	60	XXXXXXXX(1)
61	"19(1)	62	XXXXXXXX(1)	63	"20(1)	64	XXXXXXXX(1)
65	"21(1)	66	XXXXXXXX(1)	67	"22(1)	68	XXXXXXXX(1)
69	"23(1)	70	XXXXXXXX(1)	71	"24(1)	72	XXXXXXXX(1)
73	"25(1)	74	"26(1)	75	"27(1)	76	"28(1)
77	"29(1)	78	XXXXXXXX(1)	79	"30(1)	80	XXXXXXXX(1)
81	"31(1)	82	"32(1)	83	"33(1)	84	"34(1)
85	"35(1)	86	MBR(0)	87	MBR(1)	88	MBR(2)
89	MBR(3)	90	MBR(4)	91	MBR(5)	92	MBR(6)
93	MBR(7)	94	MBR(8)	95	MBR(9)	96	MBR(10)
97	MBR(11)	98	XXXXXXXX(1)	99	XXXXXXXX(1)	100	XXXXXXXX(1)
101	XXXXXXXX(1)	102	"36(1)	103	"37(1)	104	"38(1)

Figure 22(d): Synthesis Output

105	"39(	1) 106	"40(	1) 107	"41(	1) 108	"42(	1)
109	"43(	1) 110	"44(	1) 111	"45(	1) 112	"46(	1)
113	"47(	1) 114	"48(	1) 115	XXXXXXXX(	1) 116	C(	1)
117	X"1(	1) 118	C(	2) 119	X"1(	2) 120	C(	3)
121	X"1(	3) 122	C(	4) 123	X"1(	4) 124	C(	5)
125	X"1(	5) 126	C(	6) 127	X"1(	6) 128	C(	7)
129	X"1(	7) 130	C(	8) 131	X"1(	8) 132	CNTUP(	1)
133	CNTUP(	2) 134	CNTUP(	3) 135	CNTUP(	4) 136	CNTUP(	5)
137	CNTUP(	6) 138	CNTUP(	7) 139	CNTUP(	8) 140	COUT(	1)
141	X"2(	1) 142	Y(	1) 143	COUT(	2) 144	XXXXXXXX(	1)
145	X"2(	2) 146	Y(	2) 147	COUT(	3) 148	XXXXXXXX(	1)
149	X"2(	3) 150	Y(	3) 151	COUT(	4) 152	XXXXXXXX(	1)
153	X"2(	4) 154	Y(	4) 155	COUT(	5) 156	XXXXXXXX(	1)
157	X"2(	5) 158	Y(	5) 159	COUT(	6) 160	XXXXXXXX(	1)
161	X"2(	6) 162	Y(	6) 163	COUT(	7) 164	XXXXXXXX(	1)
165	X"2(	7) 166	Y(	7) 167	COUT(	8) 168	XXXXXXXX(	1)
169	X"2(	8) 170	Y(	8) 171	COUT(	9) 172	XXXXXXXX(	1)
173	X"2(	9) 174	Y(	9) 175	COUT(	10) 176	XXXXXXXX(	1)
177	X"2(	10) 178	Y(	10) 179	COUT(	11) 180	XXXXXXXX(	1)
181	X"2(	11) 182	Y(	11) 183	COUT(	12) 184	XXXXXXXX(	1)
185	X"2(	12) 186	Y(	12) 187	SUM(	1) 188	XXXXXXXX(	1)
189	SUM(	2) 190	XXXXXXXX(	1) 191	SUM(	3) 192	XXXXXXXX(	1)
193	SUM(	4) 194	XXXXXXXX(	1) 195	SUM(	5) 196	XXXXXXXX(	1)
197	SUM(	6) 198	XXXXXXXX(	1) 199	SUM(	7) 200	XXXXXXXX(	1)
201	SUM(	8) 202	XXXXXXXX(	1) 203	SUM(	9) 204	XXXXXXXX(	1)
205	SUM(	10) 206	XXXXXXXX(	1) 207	SUM(	11) 208	XXXXXXXX(	1)
209	SUM(	12) 210	XXXXXXXX(	1) 211	R(	1) 212	XXXXXXXX(	1)

Figure 22(d): (Continued)

213	XXXXXXXX(	1)	214	XXXXXXXX(	1)	215	XXXXXXXX(	1)	216	ACC(	0)
217	X(	0)	218	XXXXXXXX(	1)	219	XXXXXXXX(	1)	220	XXXXXXXX(	1)
221	XXXXXXXX(	1)	222	XXXXXXXX(	1)	223	ACC(	1)	224	X(	1)
225	XXXXXXXX(	1)	226	XXXXXXXX(	1)	227	XXXXXXXX(	1)	228	ACC(	2)
229	X(	2)	230	XXXXXXXX(	1)	231	XXXXXXXX(	1)	232	XXXXXXXX(	1)
233	ACC(	3)	234	X(	3)	235	XXXXXXXX(	1)	236	XXXXXXXX(	1)
237	XXXXXXXX(	1)	238	ACC(	4)	239	X(	4)	240	XXXXXXXX(	1)
241	XXXXXXXX(	1)	242	XXXXXXXX(	1)	243	ACC(	5)	244	X(	5)
245	XXXXXXXX(	1)	246	XXXXXXXX(	1)	247	XXXXXXXX(	1)	248	ACC(	6)
249	X(	6)	250	XXXXXXXX(	1)	251	XXXXXXXX(	1)	252	XXXXXXXX(	1)
253	ACC(	7)	254	X(	7)	255	XXXXXXXX(	1)	256	XXXXXXXX(	1)
257	XXXXXXXX(	1)	258	XXXXXXXX(	1)	259	XXXXXXXX(	1)	260	ACC(	8)
261	X(	8)	262	XXXXXXXX(	1)	263	XXXXXXXX(	1)	264	XXXXXXXX(	1)
265	XXXXXXXX(	1)	266	ACC(	9)	267	X(	9)	268	XXXXXXXX(	1)
269	XXXXXXXX(	1)	270	XXXXXXXX(	1)	271	ACC(	10)	272	X(	10)
273	XXXXXXXX(	1)	274	XXXXXXXX(	1)	275	XXXXXXXX(	1)	276	ACC(	11)
277	X(	11)	278	XXXXXXXX(	1)	279	XXXXXXXX(	1)	280	XXXXXXXX(	1)
281	MAR(	0)	282	PC(	0)	283	XXXXXXXX(	1)	284	XXXXXXXX(	1)
285	XXXXXXXX(	1)	286	XXXXXXXX(	1)	287	XXXXXXXX(	1)	288	XXXXXXXX(	1)
289	XXXXXXXX(	1)	290	XXXXXXXX(	1)	291	MAR(	1)	292	PC(	1)
293	XXXXXXXX(	1)	294	XXXXXXXX(	1)	295	XXXXXXXX(	1)	296	XXXXXXXX(	1)
297	MAR(	2)	298	PC(	2)	299	XXXXXXXX(	1)	300	XXXXXXXX(	1)
301	XXXXXXXX(	1)	302	XXXXXXXX(	1)	303	MAR(	3)	304	PC(	3)
305	XXXXXXXX(	1)	306	XXXXXXXX(	1)	307	XXXXXXXX(	1)	308	MAR(	4)
309	PC(	4)	310	IR(	4)	311	XXXXXXXX(	1)	312	XXXXXXXX(	1)
313	XXXXXXXX(	1)	314	MAR(	5)	315	PC(	5)	316	IR(	5)
317	XXXXXXXX(	1)	318	XXXXXXXX(	1)	319	XXXXXXXX(	1)	320	MAR(	6)
321	PC(	6)	322	IR(	6)	323	XXXXXXXX(	1)	324	XXXXXXXX(	1)
325	XXXXXXXX(	1)	326	XXXXXXXX(	1)	327	MAR(	7)	328	PC(	7)
329	IR(	7)	330	XXXXXXXX(	1)	331	XXXXXXXX(	1)	332	XXXXXXXX(	1)

Figure 22(d): (Continued)

```
333   XXXXXXXX(   1)  334   XXXXXXXX(   1)  335   XXXXXXXX(   1)  336   XXXXXXXX(   1)
337   XXXXXXXX(   1)  338   XXXXXXXX(   1)  339   XXXXXXXX(   1)  340   XXXXXXXX(   1)
341   XXXXXXXX(   1)  342   XXXXXXXX(   1)  343   XXXXXXXX(   1)  344   XXXXXXXX(   1)
345   XXXXXXXX(   1)  346   XXXXXXXX(   1)  347   XXXXXXXX(   1)  348   XXXXXXXX(   1)
349   XXXXXXXX(   1)  350   XXXXXXXX(   1)  351   XXXXXXXX(   1)  352   XXXXXXXX(   1)
353   XXXXXXXX(   1)  354   XXXXXXXX(   1)  355   XXXXXXXX(   1)  356   XXXXXXXX(   1)
357   XXXXXXXX(   1)  358   XXXXXXXX(   1)  359   XXXXXXXX(   1)  360   XXXXXXXX(   1)
361   XXXXXXXX(   1)  362   XXXXXXXX(   1)  363   XXXXXXXX(   1)  364   XXXXXXXX(   1)
365   XXXXXXXX(   1)  366   XXXXXXXX(   1)  367   XXXXXXXX(   1)  368   XXXXXXXX(   1)
369   XXXXXXXX(   1)  370   XXXXXXXX(   1)  371   XXXXXXXX(   1)  372   XXXXXXXX(   1)
373   XXXXXXXX(   1)  374   XXXXXXXX(   1)  375   XXXXXXXX(   1)  376   XXXXXXXX(   1)
377   XXXXXXXX(   1)  378   XXXXXXXX(   1)  379   XXXXXXXX(   1)  380   XXXXXXXX(   1)
381   XXXXXXXX(   1)  382   XXXXXXXX(   1)  383   XXXXXXXX(   1)  384   XXXXXXXX(   1)
385   XXXXXXXX(   1)  386   XXXXXXXX(   1)  387   XXXXXXXX(   1)  388   XXXXXXXX(   1)
389   XXXXXXXX(   1)  390   XXXXXXXX(   1)  391   XXXXXXXX(   1)  392   XXXXXXXX(   1)
393   XXXXXXXX(   1)  394   XXXXXXXX(   1)  395   XXXXXXXX(   1)  396   XXXXXXXX(   1)
397   XXXXXXXX(   1)  398   XXXXXXXX(   1)  399   XXXXXXXX(   1)  400   XXXXXXXX(   1)
401   XXXXXXXX(   1)  402   XXXXXXXX(   1)  403   XXXXXXXX(   1)  404   XXXXXXXX(   1)
405   XXXXXXXX(   1)  406   XXXXXXXX(   1)  407   XXXXXXXX(   1)  408   XXXXXXXX(   1)
409   XXXXXXXX(   1)  410   XXXXXXXX(   1)  411   XXXXXXXX(   1)  412   XXXXXXXX(   1)
413   XXXXXXXX(   1)  414   XXXXXXXX(   1)  415   XXXXXXXX(   1)  416   XXXXXXXX(   1)
417   XXXXXXXX(   1)  418   XXXXXXXX(   1)  419       MEAC(   1)  420   XXXXXXXX(   1)
421   XXXXXXXX(   1)  422       MBUS(   1)  423   XXXXXXXX(   1)  424       MBUS(   2)
425   XXXXXXXX(   1)  426       MBUS(   3)  427   XXXXXXXX(   1)  428       MBUS(   4)
429   XXXXXXXX(   1)  430       MBUS(   5)  431   XXXXXXXX(   1)  432       MBUS(   6)
433   XXXXXXXX(   1)  434   XXXXXXXX(   1)  435       MBUS(   7)  436   XXXXXXXX(   1)
437       MBUS(   8)  438   XXXXXXXX(   1)  439   XXXXXXXX(   1)  440       MBUS(   9)
441   XXXXXXXX(   1)  442   XXXXXXXX(   1)  443       MBUS(  10)  444   XXXXXXXX(   1)
445       MBUS(  11)  446   XXXXXXXX(   1)  447       MBUS(  12)  448   XXXXXXXX(   1)
```

Figure 22(d): (Continued)

```
449  XXXXXXXX(  1) 450  XXXXXXXX(  1) 451  XXXXXXXX(  1) 452  XXXXXXXX(  1)
453  XXXXXXXX(  1) 454  XXXXXXXX(  1) 455  XXXXXXXX(  1) 456  XXXXXXXX(  1)
457  XXXXXXXX(  1) 458  XXXXXXXX(  1) 459  XXXXXXXX(  1) 460  XXXXXXXX(  1)
461  XXXXXXXX(  1) 462  XXXXXXXX(  1) 463  XXXXXXXX(  1) 464  XXXXXXXX(  1)
465        IH(  8) 466  XXXXXXXX(  1) 467  XXXXXXXX(  1) 468        IH(  9)
469  XXXXXXXX(  1) 470  XXXXXXXX(  1) 471        IH( 10) 472  XXXXXXXX(  1)
473  XXXXXXXX(  1) 474        IH( 11) 475  XXXXXXXX(  1) 476  XXXXXXXX(  1)
477  XXXXXXXX(  1) 478  XXXXXXXX(  1) 479  XXXXXXXX(  1) 480  XXXXXXXX(  1)
481  XXXXXXXX(  1) 482  XXXXXXXX(  1) 483  XXXXXXXX(  1) 484  XXXXXXXX(  1)
485  XXXXXXXX(  1) 486  XXXXXXXX(  1) 487  XXXXXXXX(  1) 488  XXXXXXXX(  1)
489  XXXXXXXX(  1) 490  XXXXXXXX(  1) 491  XXXXXXXX(  1) 492  XXXXXXXX(  1)
493  XXXXXXXX(  1) 494  XXXXXXXX(  1) 495  XXXXXXXX(  1) 496  XXXXXXXX(  1)
497  XXXXXXXX(  1) 498  XXXXXXXX(  1) 499     AH   (  1) 500  XXXXXXXX(  1)
501  XXXXXXXX(  1) 502  XXXXXXXX(  1) 503  XXXXXXXX(  1) 504  XXXXXXXX(  1)
505  XXXXXXXX(  1) 506  XXXXXXXX(  1) 507  XXXXXXXX(  1) 508  XXXXXXXX(  1)
509  XXXXXXXX(  1) 510  XXXXXXXX(  1) 511  XXXXXXXX(  1) 512  XXXXXXXX(  1)
513  XXXXXXXX(  1) 514  XXXXXXXX(  1) 515  XXXXXXXX(  1) 516  XXXXXXXX(  1)
517  XXXXXXXX(  1) 518  XXXXXXXX(  1) 519  XXXXXXXX(  1) 520  XXXXXXXX(  1)
521  XXXXXXXX(  1) 522  XXXXXXXX(  1) 523  XXXXXXXX(  1) 524  XXXXXXXX(  1)
525  XXXXXXXX(  1) 526  XXXXXXXX(  1) 527  XXXXXXXX(  1) 528  XXXXXXXX(  1)
529  XXXXXXXX(  1) 530  XXXXXXXX(  1) 531  XXXXXXXX(  1) 532  XXXXXXXX(  1)
533  XXXXXXXX(  1) 534  XXXXXXXX(  1) 535  XXXXXXXX(  1) 536  XXXXXXXX(  1)
537  XXXXXXXX(  1) 538  XXXXXXXX(  1) 539  XXXXXXXX(  1) 540  XXXXXXXX(  1)
541  XXXXXXXX(  1) 542  XXXXXXXX(  1) 543  XXXXXXXX(  1) 544  XXXXXXXX(  1)
545  XXXXXXXX(  1) 546  XXXXXXXX(  1) 547  XXXXXXXX(  1) 548  XXXXXXXX(  1)
549  XXXXXXXX(  1) 550  XXXXXXXX(  1) 551  XXXXXXXX(  1) 552  XXXXXXXX(  1)
553  XXXXXXXX(  1) 554  XXXXXXXX(  1) 555  XXXXXXXX(  1) 556  XXXXXXXX(  1)
557  XXXXXXXX(  1) 558  XXXXXXXX(  1) 559  XXXXXXXX(  1) 560  XXXXXXXX(  1)
561  XXXXXXXX(  1) 562  XXXXXXXX(  1) 563  XXXXXXXX(  1) 564  XXXXXXXX(  1)
565  XXXXXXXX(  1) 566  XXXXXXXX(  1) 567  XXXXXXXX(  1) 568  XXXXXXXX(  1)
569  XXXXXXXX(  1)
```

Figure 22(d): (Continued)

CELL TABLE

CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL	CELL NO.	STD. CELL
1	1310	2	1310	3	1620	4	1620	5	1620	6	1620	7	1310
8	1310	9	1310	10	1310	11	1640	12	1640	13	1640	14	1640
15	1640	16	1640	17	1640	18	1640	19	1640	20	1640	21	1620
22	1620	23	1620	24	1620	25	1620	26	1620	27	1620	28	1620
29	1640	30	1230	31	1620	32	1620	33	1310	34	1620	35	1620
36	1620	37	1620	38	1620	39	1620	40	1310	41	1310	42	1310
43	1630	44	1620	45	1630	46	1620	47	1630	48	1620	49	1630
50	1620	51	1630	52	1620	53	1630	54	1620	55	1630	56	1620
57	1620	58	1620	59	1620	60	1620	61	1310	62	1620	63	1310
64	1620	65	1620	66	1620	67	1620	68	1620	69	1740	70	1740
71	1740	72	1130	73	1620	74	1620	75	1620	76	1620	77	1620
78	1620	79	1620	80	1620	81	1620	82	1620	83	1620	84	1620
85	1620	86	1620	87	1300	88	1620	89	1620	90	1620	91	1620
92	1620	93	1620	94	1620	95	2310	96	2310	97	2310	98	2310
99	2310	100	2310	101	2310	102	1310	103	1890	104	1310	105	1890
106	1310	107	1890	108	1310	109	1890	110	1310	111	1890	112	1310
113	1890	114	1310	115	1890	116	1310	117	1890	118	1310	119	1890
120	1310	121	1890	122	1310	123	1890	124	1310	125	1620	126	2310
127	2310	128	2310	129	2310	130	2310	131	2310	132	2310	133	2310
134	2310	135	2310	136	2310	137	2310	138	2310	139	2310	140	2310
141	2310	142	2310	143	2310	144	2310	145	2310	146	2310	147	2310
148	2310	149	1310	150	1800	151	1310	152	1720	153	1830	154	1720
155	1830	156	1800	157	1310	158	1630	159	1870	160	1310	161	1830

Figure 22(d): (Continued)

162	1630	163	1870	164	1310	165	1830	166	1630	167	1870	168	1310
169	1830	170	1630	171	1870	172	1310	173	1830	174	1630	175	1870
176	1310	177	1830	178	1630	179	1870	180	1310	181	1830	182	1630
183	1870	184	1310	185	1830	186	1630	187	1300	188	1870	189	1300
190	1310	191	1830	192	1630	193	1870	194	1310	195	1830	196	1300
197	1630	198	1870	199	1310	200	1830	201	1630	202	1870	203	1310
204	1830	205	1630	206	1870	207	1310	208	1830	209	1800	210	1300
211	1800	212	1220	213	1800	214	1870	215	1300	216	1220	217	1830
218	1800	219	1870	220	1300	221	1220	222	1830	223	1800	224	1300
225	1870	226	1220	227	1830	228	1800	229	1870	230	1220	231	1830
232	1800	233	1870	234	1220	235	1830	236	1800	237	1870	238	1220
239	1830	240	1800	241	1300	242	1870	243	1220	244	1830	245	1800
246	1300	247	1300	248	1300	249	1870	250	1300	251	1300	252	1220
253	1830	254	1870	255	1300	256	1310	257	1620	258	1830	259	1620
260	1830	261	1620	262	1830	263	1620	264	1830	265	1620	266	1830
267	1620	268	1830	269	1620	270	1830	271	1620	272	1300	273	1830
274	1620	275	1830	276	1300	277	1620	278	1830	279	1620	280	1830
281	1620	282	1830	283	1870	284	1310	285	1830	286	1800	287	1800
288	1300	289	1800	290	1800	291	1300	292	1870	293	1240	294	1740
295	1740	296	1740	297	1730	298	1830	299	1740	300	1310	301	1720
302	1830	303	1740	304	1310	305	1720	306	1830	307	1720	308	1830
309	1800	310	1300	311	1620	312	1220	313	1800	314	1620	315	1220
316	1830	317	1800	318	1620	319	1220	320	1830	321	1800	322	1620
323	1220	324	1830	325	1800	326	1220	327	1220	328	1830	329	1800
330	1620	331	1220	332	1830	333	1800	334	1300	335	1300	336	1620
337	1220	338	1830	339	1800	340	1620	341	1220	342	1830	343	1860
344	1300	345	1300	346	1620	347	1300	348	1220	349	1830	350	1870
351	1310	352	1870	353	1310	354	1870	355	1310	356	1870	357	1310
358	1870	359	1310	360	1870	361	1310	362	1870	363	1310	364	1870

Figure 22(d): (Continued)

NET TABLE

NET	CELL	PIN	CELL	PIN	CELL	FIN	CELL	FIN	CELL	PIN	CELL	PIN	CELL	PIN	CELL	FIN	CELL	PIN	CELL	F
1	3	4	21	3																
2	1	2	5	3	6	3	153	4												
3	2	2	4	2	6	2	155	4												
4	1	3	3	3	4	3														
5	2	3	3	2	5	2														
6	4	4	150	6	154	3														
7	5	4	150	4	152	2	154	2												
8	6	4	150	2																
9	11	6	22	3																
10	7	2	19	5	20	5	298	4												
11	8	2	15	4	16	4	17	4	18	4	302	4								
12	9	2	13	3	14	3	17	3	18	3	306	4								
13	10	2	12	2	14	2	16	2	18	2	20	2	308	4						
14	7	3	11	5	12	5	13	5	14	5	15	5	16	5	17	5	18	5		
15	8	3	11	4	12	4	13	4	14	4	19	4	20	4						
16	9	3	11	3	12	3	15	3	16	3	19	3	20	3						
17	10	3	11	2	13	2	15	2	17	2	19	2								
18	12	6	24	3																
19	13	6	35	3	36	3	37	3	38	3										
20	14	6	39	3																
21	15	6	57	3	58	3	59	3	60	3										
22	16	6	65	3	66	3	67	3	68	3										
23	17	6	74	3	75	3	76	3	77	3										
24	18	6	78	3	79	3	80	3	81	3										
25	19	6	82	3	83	3	84	3												
26	20	6	85	3	87	3														
27																				
28	21	2	22	2																
29	22	4	23	3																
30	23	4	156	8	210	8	213	9	218	9	224	9	228	9	232	9	236	9	241	
31	23	2	28	2	38	2	39	2	60	2	68	2	77	2	81	2	84	2	87	
32	24	4	25	3	26	3	27	3	28	3										
33	24	2	285	4																
34	25	4	210	6	213	7	218	7	224	7	228	7	232	7	236	7	240	7	248	
35	25	2	35	2	57	2	65	2	74	2	78	2	82	2	85	2				
36	26	6	309	8	313	9	317	9	321	9	325	9	329	9	335	9	339	9	345	
37	26	2	36	2	58	2	66	2	75	2	79	2	83	2						
38	27	4	398	2	399	3	401	3	403	3	405	3	407	4	409	5	412	5	416	
39	27	2	37	2	59	2	67	2	76	2	80	2								
40	28	4	29	5	31	3														
41	29	6	283	2	286	6														
42	29	4	30	4	40	2	51	4	53	4	55	4	213	2	213	4	214	2	215	
43	29	3	30	3	41	2	47	3	49	3	55	3	218	2	218	4	219	2	220	
44	29	2	30	2	42	2	45	2	49	2	53	2	61	2	64	2	223	2	224	
45	31	4	32	3	34	3														
46	30	5	31	2																
47	32	4	286	4	299	5														
48	32	2	33	2	228	2	228	4	229	2	229	4	325	4	326	2	406	4		
49	34	4	286	2	294	4	299	4												
50	33	3	34	2																
51	35	4	209	4	213	5	218	5	224	5	228	5	232	5	236	5	240	5	247	
52	36	4	368	4	486	4	491	7	496	7	502	7	507	7	512	4	518	7	526	
53	37	4	407	2	409	3	412	3	415	3	419	3	422	3	425	3	428	3	432	

Figure 22(d): (Continued)

54	38	4	288	8	294	3	299	3											
55	39	4	44	3	46	3	48	3	50	3	52	3	54	3	56	3			
56	44	4	288	6	303	5													
57	40	3	43	4	45	4	47	4	49	4									
58	41	3	43	3	45	3	51	3	53	3									
59	42	3	43	2	47	2	51	2	55	2									
60	43	5	44	2															
61	46	4	287	4	303	4													
62	45	5	46	2															
63	48	4	287	2	294	2	303	3											
64	47	5	48	2															
65	50	4	289	8	299	2	303	2											
66	49	5	50	2															
67	52	4	289	6	295	5	300	2	304	2									
68	51	5	52	2															
69	54	4	289	4	295	4	307	3											
70	53	5	54	2															
71	56	4	289	2	307	2													
72	55	5	56	2															
73	57	4	254	2	257	3	259	3	261	3	263	3	265	3	267	3	269	3	272	2
74	58	4	209	2	213	3	216	3	223	3	228	3	232	3	236	3	240	3	246	2
75	59	4	368	3	485	2	491	5	496	5	502	5	507	5	512	2	518	5	525	2
76	60	4	62	3	64	3	291	8	295	3									
77	62	4	156	6	158	4	162	4	166	4	170	4	174	4	178	4	182	4	187	2
78	61	3	62	2															
79	64	4	156	4	159	5	163	5	167	5	171	5	175	5	179	5	183	5	189	2
80																			
81	65	4	211	8	215	5	220	5	225	5	229	5	233	5	237	5	242	5	251	2
82	66	4	368	2	487	8	491	3	496	3	502	3	507	3	514	8	518	3	524	2
83	67	4	435	3	437	3	439	3	441	3	444	3	447	3	450	3	453	3	457	2
84	68	4	73	3	291	6	295	2	371	7	373	7	375	7	377	7	379	7	382	2
85	73	4	309	6	313	7	317	7	321	7	325	7	329	7	335	7	339	7	344	6
86	69	5	158	3	313	2	371	2	371	4	371	6	399	2	435	2	435	4	495	2
87	69	4	162	3	317	2	373	2	373	4	373	6	401	2	437	2	437	4	501	2
88	69	3	166	3	321	2	375	2	375	4	375	6	403	2	439	2	439	4	506	2
89	69	2	170	3	325	2	377	2	377	4	377	6	405	2	441	2	441	4	511	6
90	70	5	174	3	329	2	379	2	379	4	379	6	409	2	409	4	444	2	445	2
91	70	4	178	3	333	2	381	2	381	4	381	6	412	2	412	4	447	2	448	2
92	70	3	182	3	339	2	384	2	384	4	384	6	415	2	415	4	450	2	451	2
93	70	2	186	3	343	2	386	2	386	4	387	6	419	2	419	4	453	2	454	2
94	71	5	192	3	389	2	390	4	390	6	422	2	422	4	456	2	457	4	546	2
95	71	4	197	3	392	2	392	4	392	6	425	2	425	4	459	2	459	4	551	2
96	71	3	201	3	394	2	394	4	394	6	428	2	428	4	461	2	461	4	557	2
97	71	2	205	3	396	2	396	4	396	6	431	2	432	4	463	2	463	4	562	2
98	69	6	72	2															
99	70	6	72	3															
100	71	6	72	4	73	2													
101	72	5																	
102	74	4	487	4	492	5	497	5	503	5	508	5	514	4	520	7	528	7	533	6
103	75	4	211	6	214	3	219	3	225	3	229	3	233	3	237	3	242	3	250	2
104	76	4	156	2	371	5	373	5	375	5	377	5	379	5	381	5	384	5	387	2
105	77	4	290	4	296	5													
106	78	4	488	2	516	2	520	3	527	3	532	3	537	3	542	3	548	3	554	2
107	79	4	211	4															
108	80	4	371	3	373	3	375	3	377	3	379	3	381	3	384	3	386	3	390	2
109	81	4	290	2	296	4	309	4	313	5	317	5	321	5	325	5	329	5	335	2
110	82	4	211	2															
111	83	4	369	2	487	2	492	3	497	3	503	3	508	3	514	2	520	5	528	2
112	84	4	292	4	296	3	309	2	313	3	317	3	321	3	325	3	329	3	334	6

Figure 22(d): (Continued)

113	85	4	311	2	314	3	318	3	322	3	326	3	330	3	336	3	340	3	347	
114	87	4	292	2	296	2														
115	87	3	87	2	21	4														
116	88	4																		
117	88	3	95	2	351	3														
118	88	2	89	4	95	3														
119	89	3	96	2	353	3														
120	89	2	90	4	96	3														
121	90	3	97	2	355	3														
122	90	2	91	4	97	3														
123	91	3	98	2	357	3														
124	91	2	92	4	98	3														
125	92	3	99	2	359	3														
126	92	2	93	4	99	3														
127	93	3	100	2	361	3														
128	93	2	94	4	100	3														
129	94	3	101	2	363	3														
130	94	2	101	3	102	2														
131																				
132	95	4	313	6	313	8														
133	96	4	317	6	317	8														
134	97	4	321	6	321	8														
135	98	4	325	6	325	8														
136	99	4	329	6	329	8														
137	100	4	335	6	335	8														
138	101	4	339	6	339	8														
139	102	3	343	6	344	8														
140	104	3																		
141	103	5	103	7	126	2	436	3												
142	103	3	103	6	126	3	467	3												
143	103	2	103	4	106	3	127	3												
144	103	8	104	2																
145	105	5	105	7	128	2	438	3												
146	105	3	105	6	128	3	468	4												
147	105	2	105	4	108	3	129	3												
148	105	8	106	2																
149	107	5	107	7	130	2	440	3												
150	107	3	107	6	130	3	469	4												
151	107	2	107	4	110	3	131	3												
152	107	8	108	2																
153	109	5	109	7	132	2	443	3												
154	109	3	109	6	132	3	470	4												
155	109	2	109	4	112	3	133	3												
156	109	8	110	2																
157	111	5	111	7	134	2	446	3												
158	111	3	111	6	134	3	471	4												
159	111	2	111	4	114	3	135	3												
160	111	8	112	2																
161	113	5	113	7	136	2	449	3												
162	113	3	113	6	136	3	472	4												
163	113	2	113	4	116	3	137	3												
164	113	8	114	2																
165	115	5	115	7	138	2	452	3												
166	115	3	115	6	138	3	473	4												
167	115	2	115	4	118	3	139	3												
168	115	8	116	2																
169	117	5	117	7	140	2	455	3												
170	117	3	117	6	140	3	475	4												
171	117	2	117	4	120	3	141	3												

Figure 22(d): (Continued)

172	117	8	118	2																
173	119	5	119	7	142	2	458	3												
174	119	3	119	6	142	3	476	4												
175	119	2	119	4	122	3	143	3												
176	119	8	120	2																
177	121	5	121	7	144	2	460	3												
178	121	3	121	6	144	3	477	4												
179	121	2	121	4	124	3	145	3												
180	121	8	122	2																
181	123	5	123	7	146	2	462	3												
182	123	3	123	6	146	3	478	4												
183	123	2	123	4	125	4	147	3												
184	123	8	124	2																
185	125	3	148	2	465	3														
186	125	2	148	3	479	4														
187	127	4	159	4	492	6														
188	126	4	127	2																
189	129	4	163	4	497	6														
190	128	4	129	2																
191	131	4	167	4	503	6														
192	130	4	131	2																
193	133	4	171	4	508	6														
194	132	4	133	2																
195	135	4	175	4	520	8														
196	134	4	135	2																
197	137	4	179	4	528	8														
198	136	4	137	2																
199	139	4	183	4	533	8														
200	138	4	139	2																
201	141	4	188	4	537	8														
202	140	4	141	2																
203	143	4	193	4	542	8														
204	142	4	143	2																
205	145	4	198	4	548	8														
206	144	4	145	2																
207	147	4	202	4	554	8														
208	146	4	147	2																
209	149	3	206	4	559	8														
210	148	4	149	2																
211	150	3	150	5	150	7	150	9	156	3	156	5	156	7	156	9	209	3	210	2
212	150	10	151	2																
213	151	3	153	2	155	2														
214	152	4	153	3																
215	154	4	155	3																
216	161	4	257	2	492	4														
217	158	2	258	4	466	4														
218	156	10	157	2																
219	157	3	161	2	165	2	169	2	173	2	177	2	181	2	185	2	191	2	196	2
220	158	5	159	2	159	3														
221	159	8	160	2																
222	160	3	161	3																
223	165	4	259	2	497	4														
224	162	2	260	4	468	2														
225	162	5	163	2	163	3														
226	163	8	164	2																
227	164	3	165	3																
228	169	4	261	2	503	4														
229	166	2	262	4	469	2														
230	166	5	167	2	167	3														

Figure 22(d): (Continued)

231	167	6	168	2																
232	168	3	169	3																
233	173	4	263	2	508	4														
234	170	2	264	4	470	2														
235	170	5	171	2	171	3														
236	171	6	172	2																
237	172	3	173	3																
238	177	4	265	2	520	6														
239	174	2	266	4	471	2														
240	174	5	175	2	175	3														
241	175	6	176	2																
242	176	3	177	3																
243	181	4	267	2	528	6														
244	178	2	268	4	472	2														
245	178	5	179	2	179	3														
246	179	6	180	2																
247	180	3	181	3																
248	185	4	269	2	532	6														
249	182	2	270	4	473	2														
250	182	5	183	2	183	3														
251	183	6	184	2																
252	184	3	185	3																
253	191	4	271	2	537	6														
254	186	2	273	4	474	2														
255	187	3	187	4	192	4	197	4	201	4	205	4								
256	187	5	188	2	188	3														
257	189	3	189	5	193	5	198	5	202	5	206	5	435	5	437	5	439	5	442	2
258	189	6	190	2																
259	190	3	191	3																
260	196	4	274	2	542	6														
261	192	2	276	4	476	2														
262	192	5	193	2	193	3														
263	193	6	194	2																
264	194	3	196	3																
265	196	3	196	2	200	2	204	2	208	2										
266	200	4	277	2	548	6														
267	197	2	278	4	477	2														
268	197	5	198	2	198	3														
269	198	6	199	2																
270	199	3	200	3																
271	204	4	279	2	554	6														
272	201	2	280	4	478	2														
273	201	5	202	2	202	3														
274	202	6	203	2																
275	203	3	204	3																
276	208	4	281	2	559	6														
277	205	2	282	4	479	2														
278	205	5	206	2	206	3														
279	206	6	207	2																
280	207	3	208	3																
281	217	4																		
282	213	6	213	8	316	4	350	2	350	4	520	2	527	2	532	2	537	2	542	
283	210	3	210	5	210	7	210	9	211	3	211	5	211	7	211	9	254	3	255	
284	210	10	212	2																
285	211	10	212	3																
286	212	4	217	2	222	2	227	2	231	2	235	2	239	2	244	2	253	2		
287	213	10	216	2																
288	215	3	215	4	313	4	314	2	400	4										
289	215	6	216	3																

Figure 22(d): (Continued)

290	216	4	217	3																		
291	222	4																				
292	218	6	218	8	320	4	352	2	352	4												
293	218	10	221	2																		
294	220	3	220	4	317	4	318	2	402	4												
295	220	6	221	3																		
296	221	4	222	3																		
297	227	4																				
298	224	6	224	8	324	4	354	2	354	4												
299	224	3	224	4	225	2	225	4	321	4	322	2	404	4								
300	224	10	226	2																		
301	225	6	226	3																		
302	226	4	227	3																		
303	231	4																				
304	228	6	228	8	328	4	356	2	356	4												
305	228	10	230	2																		
306	229	6	230	3																		
307	230	4	231	3																		
308	235	4																				
309	232	6	232	8	332	4	358	2	358	4												
310	232	2	232	4	233	2	233	4	329	4	330	2	411	4								
311	232	10	234	2																		
312	233	6	234	3																		
313	234	4	235	3																		
314	239	4																				
315	236	6	236	8	338	4	360	2	360	4												
316	236	2	236	4	237	2	237	4	334	4	336	2	414	4								
317	236	10	238	2																		
318	237	6	238	3																		
319	238	4	239	3																		
320	244	4																				
321	240	6	240	8	342	4	362	2	362	4												
322	240	2	240	4	242	2	242	4	339	4	340	2	418	4								
323	241	3	241	9	248	9	254	4	283	4	285	3	286	8	294	5	486	8	513	2		
324	241	10	243	2																		
325	242	6	243	3																		
326	243	4	244	3																		
327	253	4																				
328	247	6	248	8	349	4	364	2	365	4												
329	245	2	246	4	249	2	250	4	343	4	346	2	421	4								
330	246	3	246	3																		
331	247	3	247	5																		
332	248	3	248	7																		
333	248	10	252	2																		
334	250	3	250	3																		
335	251	3	251	5																		
336	251	6	252	3																		
337	252	4	253	3																		
338	255	3	255	5	283	3	283	5	286	3	286	5	286	7	286	9	287	3	288	2		
339	255	6	256	2																		
340	256	3	256	2	260	2	262	2	264	2	266	2	268	2	270	2	273	2	276	2		
341	257	4	258	3																		
342	259	4	260	3																		
343	261	4	262	3																		
344	263	4	264	3																		
345	265	4	266	3																		
346	267	4	268	3																		
347	269	4	270	3																		
348	272	3	272	3	274	3	277	3	279	3	281	3										

Figure 22(d): (Continued)

349	272	4	273	3														
350	274	4	276	3														
351	276	3	276	2	278	2	280	2	282	2								
352	277	4	278	3														
353	279	4	280	3														
354	281	4	282	3														
355	283	6	284	2														
356	284	3	285	2														
357	286	10	293	2														
358	288	3	288	5	288	7	288	9	289	3	289	5	289	7	289	9	290	3	291
359	288	10	293	3														
360	289	10	293	4														
361	291	3	291	5	291	7	291	9	292	3	292	5	309	3	309	5	309	7	310
362	291	10	293	5														
363	292	6																
364	293	6	298	2	302	2	306	2	308	2								
365	294	6	297	2														
366	295	6	297	3														
367	296	6	297	4														
368	297	5	298	3														
369	299	6	301	2														
370	300	3	301	3														
371	301	4	302	3														
372	303	6	305	2														
373	304	3	305	3														
374	305	4	306	3														
375	307	4	308	3														
376	310	3	310	9	311	3	398	3	407	3	407	5	480	3	480	5	480	7	488
377	310	10	312	2														
378	311	4	312	3														
379	312	4	316	2	320	2	324	2	328	2	332	2	338	2	342	2	349	2	
380	313	10	315	2														
381	314	4	315	3														
382	315	4	316	3														
383	317	10	319	2														
384	318	4	319	3														
385	319	4	320	3														
386	321	10	323	2														
387	322	4	323	3														
388	323	4	324	3														
389	325	10	327	2														
390	326	4	327	3														
391	327	4	328	3														
392	329	10	331	2														
393	330	4	331	3														
394	331	4	332	3														
395	334	3	334	3	339	3	343	3										
396	335	3	335	5	339	5	343	5										
397	335	10	337	2														
398	336	4	337	3														
399	337	4	338	3														
400	339	10	341	2														
401	340	4	341	3														
402	341	4	342	3														
403	344	3	344	7	350	3	352	3	354	3	356	3	358	3	360	3	362	3	365
404	345	3	345	9	350	5	352	5	354	5	356	5	358	5	360	5	362	5	366
405	345	10	348	2														
406	347	3	347	3														
407	347	4	348	3														

Figure 22(d): (Continued)

408	348	4	349	3															
409	350	6	351	2															
410	352	6	353	2															
411	354	6	355	2															
412	356	6	357	2															
413	358	6	359	2															
414	360	6	361	2															
415	362	6	363	2															
416	365	3	365	3															
417	366	3	366	5	368	5	486	6	491	9	496	9	502	9	507	9	512	6	519
418	366	6	367	2															
419	370	4																	
420	368	8	370	2															
421	369	3	370	3															
422	372	3	491	2	491	4	491	6	491	8	492	2							
423	371	8	372																
424	374	3	496	2	496	4	496	6	496	8	497	2							
425	373	8	374	2															
426	376	3	502	2	502	4	502	6	502	8	503	2							
427	375	8	376	2															
428	378	3	507	2	507	4	507	6	507	8	508	2							
429	377	8	378	2															
430	380	3	518	2	518	4	518	6	518	8	520	4							
431	379	8	380	2															
432	383	3	523	2	524	4	525	6	526	6	527	4							
433	382	3	382	7	384	7	387	7	390	7	392	7	394	7	396	7	480	6	483
434	382	6	383	2															
435	385	3	531	2	531	4	531	6	531	8	532	4							
436	384	8	385	2															
437	388	3	536	2	536	4	536	6	536	8	537	4							
438	387	3	387	5	390	5	392	5	394	5	396	5	480	4	482	3			
439	387	8	388	2															
440	391	3	540	2	540	4	540	6	540	8	542	4							
441	390	3	390	3	392	3	394	3	396	3	480	2	482	2					
442	390	8	391	2															
443	393	3	547	2	547	4	547	6	547	8	548	4							
444	392	6	393	2															
445	345	3	552	2	552	4	552	6	552	8	554	4							
446	394	8	395	2															
447	397	3	558	2	558	4	558	6	558	8	559	4							
448	396	6	397	2															
449	398	4	400	2	402	2	404	2	406	2									
450	399	4	400	3															
451	401	4	402	3															
452	403	4	404	3															
453	405	4	406	3															
454	407	6	408	2															
455	408	3	411	2	414	2	418	2	421	2	424	2	427	2	430	2	434	2	
456	409	6	410	2															
457	410	3	411	3															
458	412	6	413	2															
459	413	3	414	3															
460	416	3	416	5	419	5	422	5	425	5	428	5	432	5					
461	416	6	417	2															
462	417	3	418	3															
463	419	6	420	2															
464	420	3	421	3															
465	424	4																	
466	422	6	423	2															

Figure 22(d): (Continued)

467	423	3	424	3																
468	427	4																		
469	425	6	426	2																
470	426	3	427	3																
471	430	4																		
472	428	6	429	2																
473	429	3	430	3																
474	434	4																		
475	432	3	432	3																
476	432	6	433	2																
477	433	3	434	3																
478	435	6	436	2																
479	437	6	438	2																
480	439	6	440	2																
481	442	3	442	5	445	5	448	5	451	5	454	5	457	5	459	5	461	5	464	2
482	442	6	443	2																
483	445	3	445	4	522	4														
484	445	6	446	2																
485	448	3	448	4	530	4														
486	448	6	449	2																
487	451	3	451	4	535	4														
488	451	6	452	2																
489	454	3	454	4	539	4														
490	454	6	455	2																
491	457	3	457	3	459	3	461	3	463	3	466	2	466	3	487	6	492	7	498	2
492	457	6	458	2																
493	459	6	460	2																
494	461	6	462	2																
495	464	3	464	5	466	5	468	3	469	3	470	3	471	3	472	3	473	3	475	2
496	464	6	465	2																
497	466	6	467	2																
498	475	3	475	3	476	3	477	3	478	3	479	3								
499	484	4																		
500	480	6	481	2																
501	481	3	484	2																
502	483	3	483	4																
503	483	5	484	3																
504	486	3	486	3	486	5	486	7	486	9	487	3	487	5	487	7	487	9	489	2
505	488	10	490	2																
506	487	10	490	3																
507	489	3	489	3	512	3	512	5	512	7	513	9	514	3	514	5	514	7	515	2
508	489	4	490	4																
509	490	5	494	2	500	2	505	2	510	2										
510	491	10	493	2																
511	492	6	493	3																
512	493	4	494	3																
513	495	3	495	4																
514	496	10	499	2																
515	498	3	498	7	503	7	508	7	514	6	520	9	528	9	533	9	537	9	543	2
516	498	6	499	3																
517	499	4	500	3																
518	501	3	501	4																
519	502	10	504	2																
520	503	6	504	3																
521	504	4	505	3																
522	506	3	506	4																
523	507	10	509	2																
524	508	6	509	3																
525	509	4	510	3																

Figure 22(d): (Continued)

526	511	3	511	4														
527	513	3	513	8														
528	513	10	517	2														
529	515	3	515	9	516	3												
530	515	10	517	3														
531	516	4	517	4														
532	517	5	522	2	530	2	535	2	539	2	545	2	550	2	556	2	561	[illegible]
533	519	3	519	9	526	9	531	9	536	9	540	9	547	9	552	9	558	[illegible]
534	519	10	521	2														
535	520	10	521	3														
536	521	4	522	3														
537	524	3	524	3	531	3	536	3	540	3	547	3	552	3	558	3		
538	525	3	525	5	531	5	536	5	540	5	547	5	552	5	558	5		
539	526	3	526	7	531	7	536	7	540	7	547	7	552	7	558	7		
540	526	10	529	2														
541	528	3	528	5	532	5	537	5	542	5	548	5	554	5	559	5		
542	528	10	529	3														
543	529	4	530	3														
544	531	10	534	2														
545	533	3	533	7	537	7	542	7	548	7	554	7	559	7				
546	533	10	534	3														
547	534	4	535	3														
548	536	10	538	2														
549	537	10	538	3														
550	538	4	539	3														
551	540	10	544	2														
552	542	3	542	2	548	2	553	2	559	2								
553	543	3	543	9	548	9	554	9	559	9								
554	543	10	544	3														
555	544	4	545	3														
556	546	3	546	4														
557	547	10	549	2														
558	548	10	549	3														
559	549	4	550	3														
560	551	3	551	4														
561	552	10	555	2														
562	554	3	554	3	559	3												
563	554	10	555	3														
564	555	4	556	3														
565	557	3	557	4														
566	558	10	560	2														
567	559	10	560	3														
568	560	4	561	3														
569	562	3	562	4														

Figure 22(d): (Continued)

Figure 22(d) shows the synthesis output for the minicomputer.

6.6 SUMMARY

From the above examples, it is clear that DDLSYN adds some extra inverters in an automatic design. Other extra cells in an automatic design are dependent on the DDL description. State declaration adds some extra cells in the automatic design. Careful use of state is absolutely necessary to keep the logic optimum.

CHAPTER 7. CONCLUSIONS

An algorithm to synthesize the logic implied by a DDL description along with its implementation details were given. The net and cell table outputs of the synthesis program DDLSYN are of the format required by the other CADAT programs. Manual generation and inputting of the net data into the CADAT system is very time consuming. DDLSYN does this automatically. The net table created by DDLSYN is stored as a disk resident file. Other CADAT programs can use this file very easily.

The DDLSYN is general enough to be useful in any LSI design environment. Formal logic minimization aspects were not considered during DDLSYN implementation, although the constants and operations involving constants were taken care of. A D flip-flop is assumed for each bit of the registers declared in DDL. The available inverted output of the flip-flop is not used in the synthesis process. Hence, some additional inverters are generated for synthesis. The modularity DDL description is lost during synthesis, since the system is synthesized as a single unit.

The hardware generated by DDLSYN is almost equivalent in cost to that designed manually. Some additional hardware is generated due to the finite state machine model assumed by DDL. Careful use of <ID> and <BO> declarations is needed to generate minimum hardware.

BIBLIOGRAPHY

1. S. G. Shiva, "Combinational Logic Synthesis from an HDL Description," Proc. of 17th Design Auto. Conf., pp. 550-555, June 1980.

2. S. G. Shiva, "Computer Hardware Description Languages - A Tutorial," Proc. IEEE, Vol. 56, pp. 1605-1615, Dec. 1979.

3. D.L. Dietmeyer and J.R. Duley, "Register Transfer Languages and their Translation," in Digital System Design Automation: Languages Simulation and Data Base, M.S. Breuer, Ed., Woodland Hills, CA: Computer Science Press, 1975, Ch. 2, pp. 117-218.

4. S. G. Shiva, "Digital Systems Design Language," NAS8-33096, Marshall Space Flight Center, AL, Oct. 1979.

5. W. M. VanCleemput, "Computer Hardware Description Languages and their Applications," Proc. Design Auto. Conf., pp. 554-560, June 1979.

6. W. M. VanCleemput, "An Hierarchial Language for the Structural Description of Digital Systems," in Proc. Design Auto. Conf., pp. 377-385, 1977.

7. T. C. Bartee, Digital Computer Fundamentals, Fourth Edition, McGraw-Hill Kogakusha, LTD., Tokyo, 1977.

8. H. Taub and D. Schilling, Digital Integrated Electronics, McGraw-Hill Kogakusha, LTD., Tokyo, 1977.

9. W. M. VanCleemput, "The Role of Design Automation in the Design of Digital Systems," Computer-Aided Design Tools for Digital Systems, IEEE Catalog No. EH0132-1, pp. 1-8, 1979.

10. S. G. Shiva, "Use of DDL in an Automatic LSI Design and Test System," Proc. Int. Conf. on CHDL's Applications, pp. 28-32, Oct. 1979.

11. S. G. Shiva, "A Comparison of Hardware Description Languages," NASA CR-157762, AL, Oct. 1978.

12. J. M. Gould, "The Large Scale Microelectronics Computer-Aided Design and Test System," NASA TM-78202, Marshall Space Flight Center, AL, Oct. 1978.

13. T. M. Edge, "C-MOS Bulk Metal Design Handbook," NASA TM-78126, Marshall Space Flight Center, AL, June 1977.

14. D. L. Dietmeyer and J. R. Duley, "A Digital Systems Design Language (DDL)," IEEE Trans. Comput., Vol. C-17, pp. 350-361, Sept. 1968.

15. D. L. Dietmeyer, "DDLTRN-Users Manual," Dept. Elec. Comput. Eng., University of Wisconsin-Madison.

16. D. L. Dietmeyer, "DDLSIM-Users Manual," Dept. Elec. Comput. Eng., University of Wisconsin-Madison.

17. D. L. Dietmeyer, "Introducing DDL," Computer Aided Design Tools for Digital Systems, IEEE Catalog No. EH0132-1, pp. 55-59, 1979.

18. M. H. Doshi and D. L. Dietmeyer, "Automated PLA Synthesis of the Combinational Logic of a DDL Description," University of Wisconsin-Madison, Rep. ECE 78-17, Nov. 1978.

19. R. E. Swanson, Z. Navabi and F. J. Hill, "An AHPL Compiler/Simulator System," Proc. Sixth Texas Conf. Comput. Syst., pp. 595-613, July 1968.

20. N. Kawato, T. Saito, F. Maruyama, and T. Uehara, "Design and Verification of Large Scale Computers by using DDL," in Proc. Design Auto. Conf., pp. 360-366, June 1979.

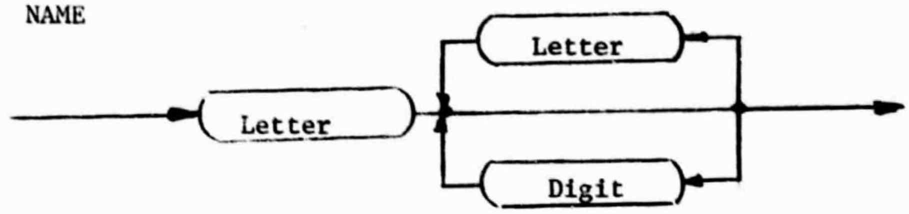
21. T. D. Friedman and S. C. Yang, "Methods used in an Automatic Logic Design Generator (ALERT)," IEEE Trans. Comput. Vol. C-18, pp. 595-613, July 1968.

APPENDIX A. SYNTAX DIAGRAMS FOR DDL CONSTRUCTS

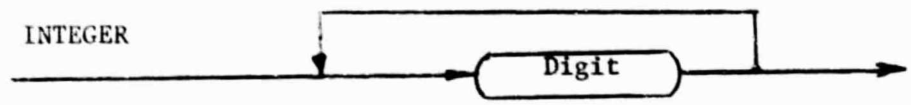
APPENDIX A. SYNTAX DIAGRAMS FOR DDL CONSTRUCTS

The syntax diagrams for the DDL description constructs are given in this appendix. For the detailed description and examples of DDL constructs, refer to [6].

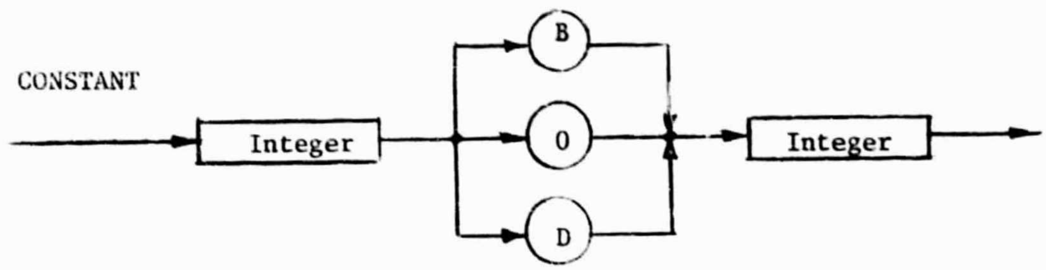
NAME



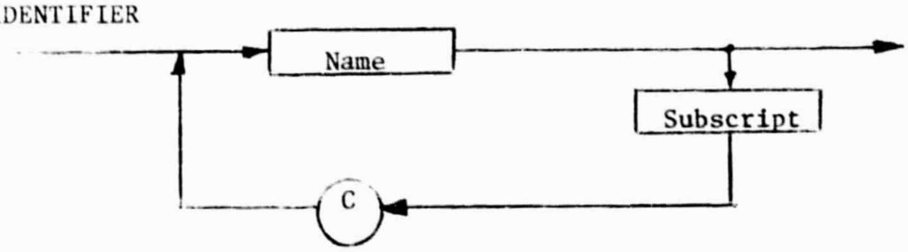
INTEGER



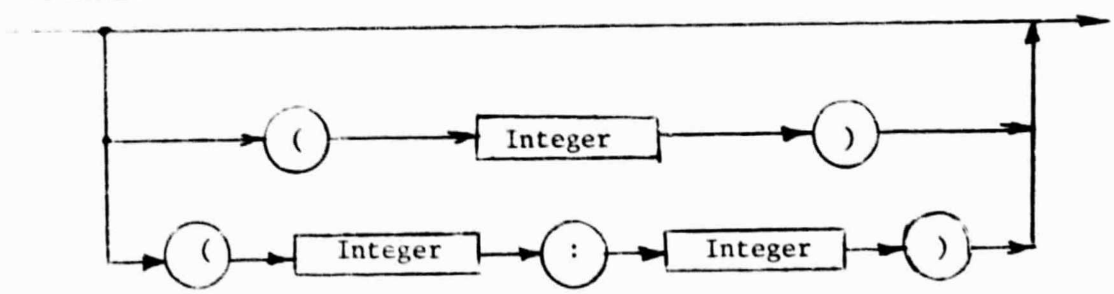
CONSTANT



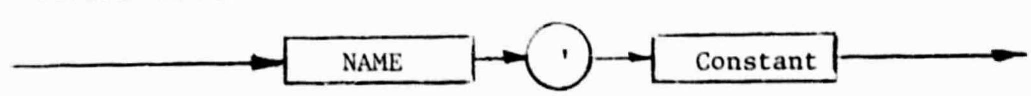
SECONDARY IDENTIFIER



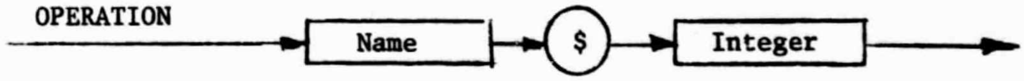
SUBSCRIPT



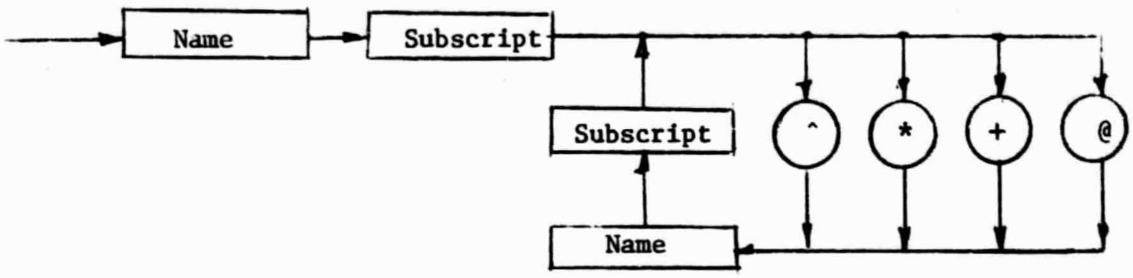
SELECTION OPERATION



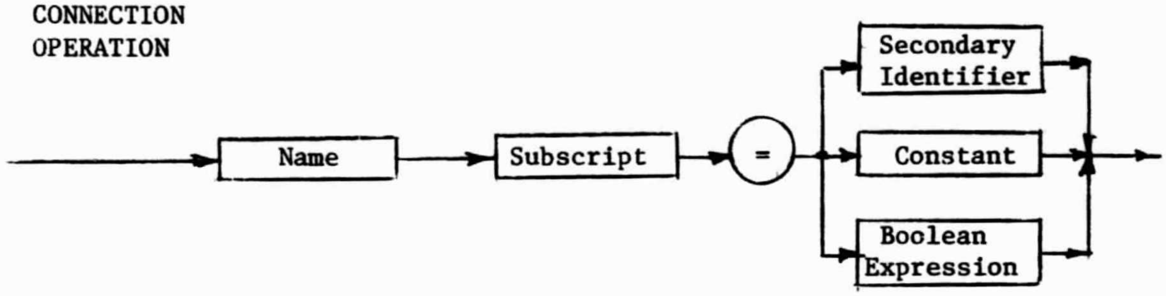
EXTENSION
OPERATION



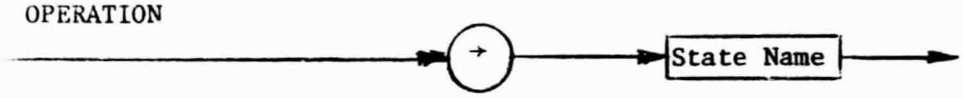
BOOLEAN EXPRESSION



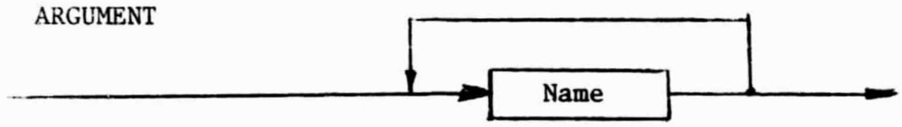
CONNECTION
OPERATION



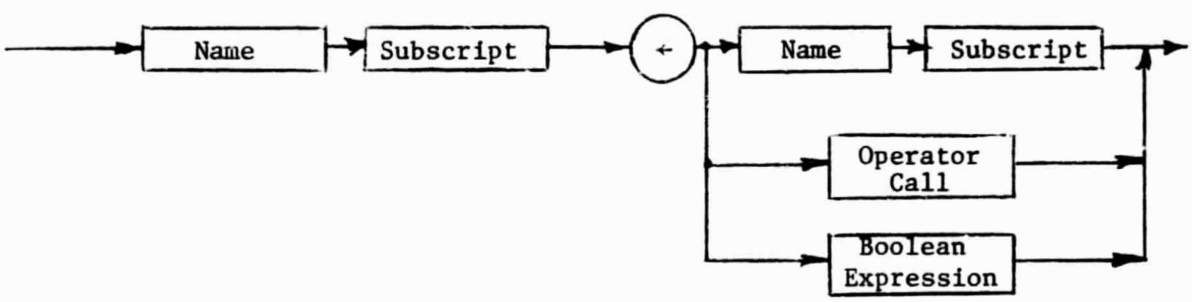
GO TO
OPERATION



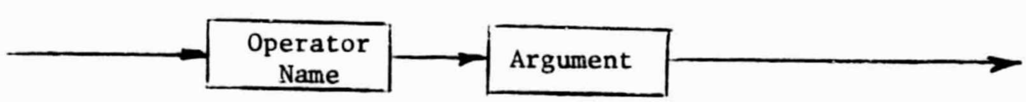
ARGUMENT



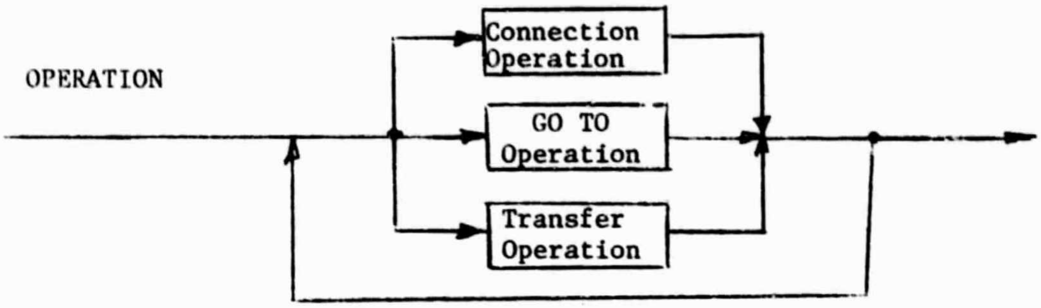
TRANSFER
OPERATOR



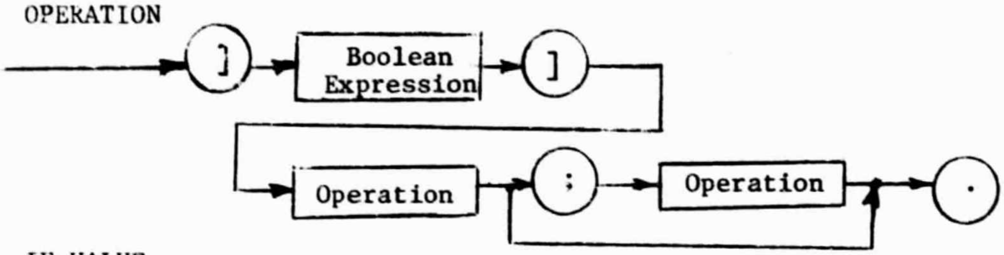
OPERATOR
CALL



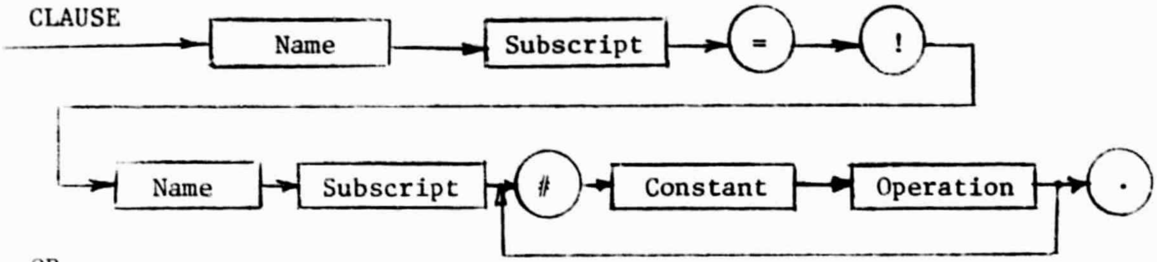
OPERATION



CONDITION
OPERATION

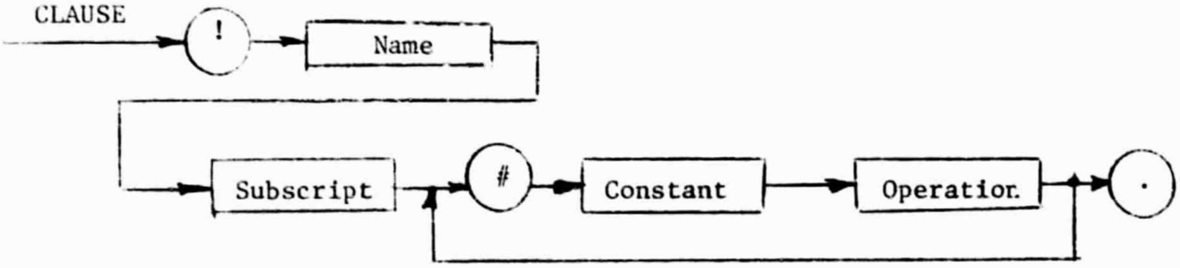


IF-VALUE
CLAUSE

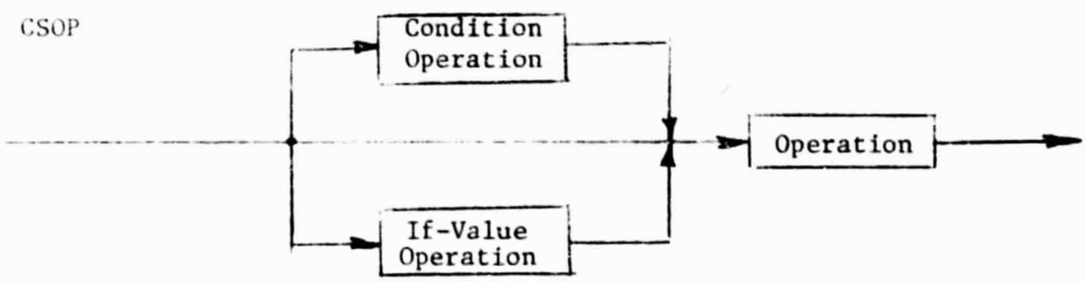


OR

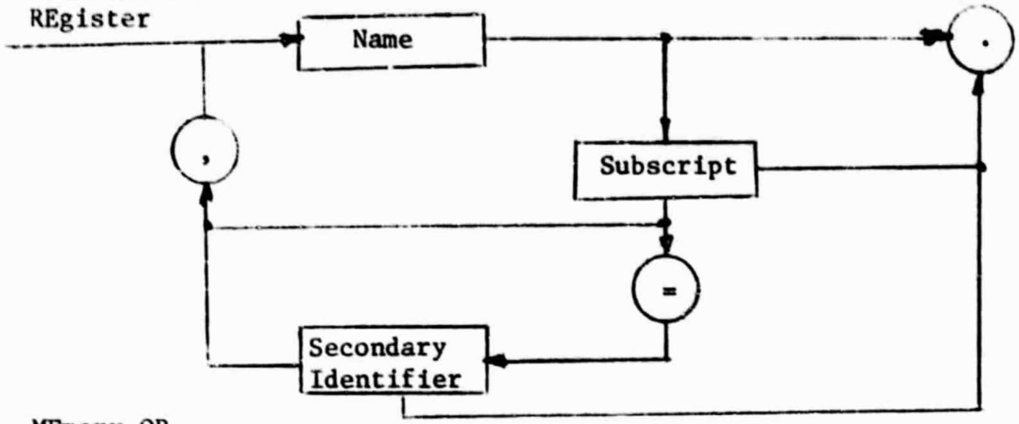
IF-VALUE
CLAUSE



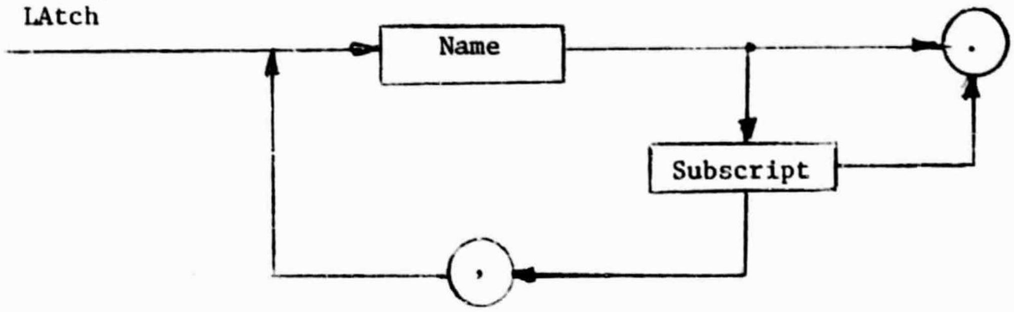
CSOP



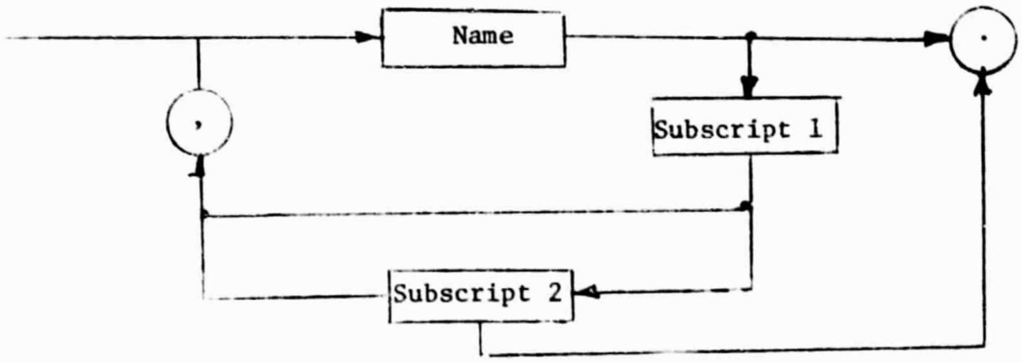
TErminal OR
REgister



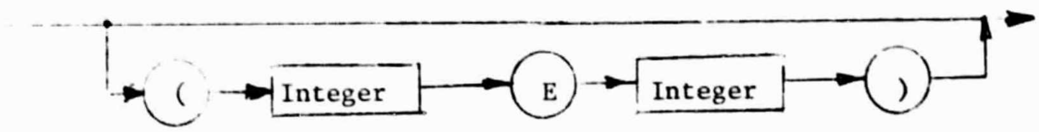
MEmory OR
LAtch



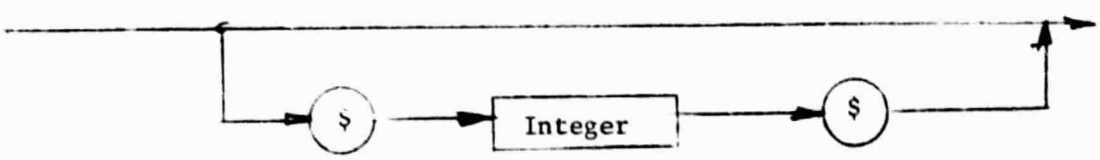
TIme OR
DElay



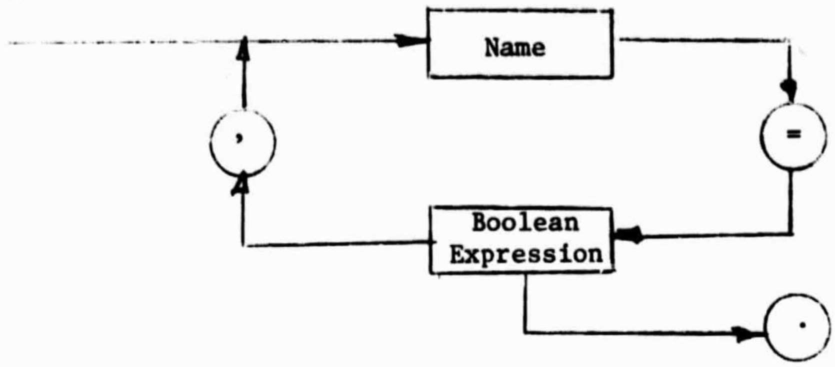
SUBSCRIPT 1



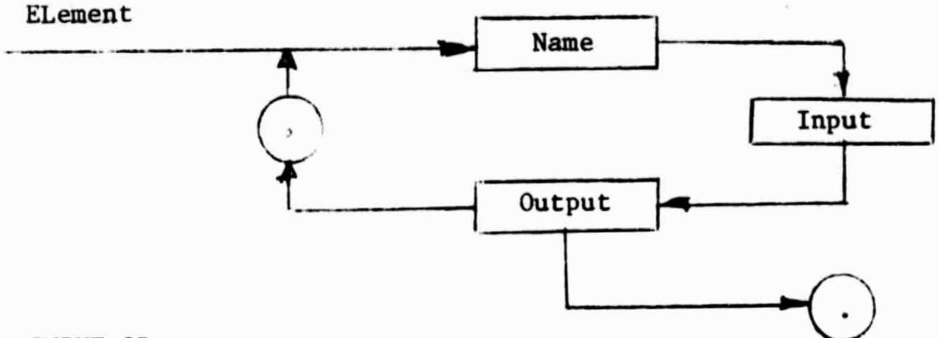
SUBSCRIPT 2



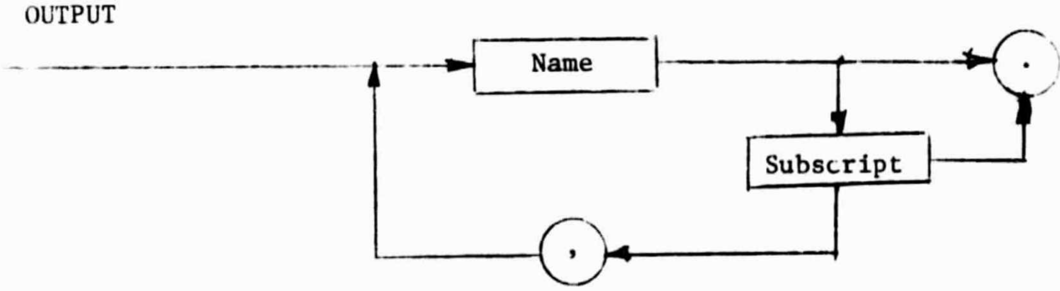
BOolean
OPERATION



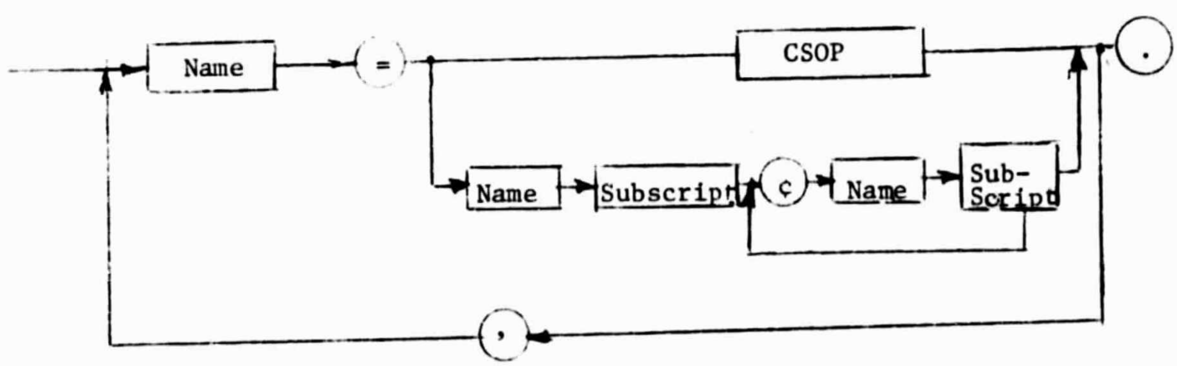
ELement



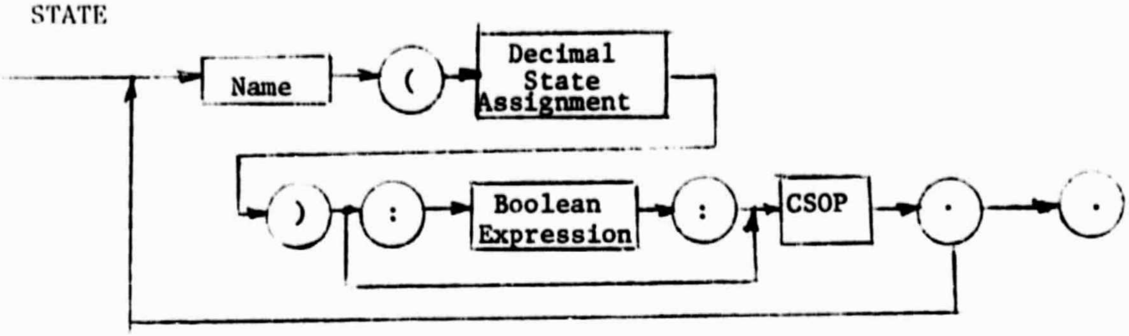
INPUT OR
OUTPUT



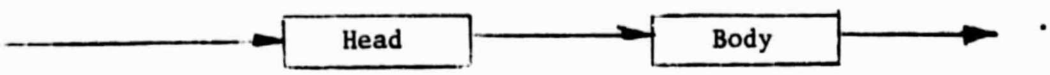
IDentifier



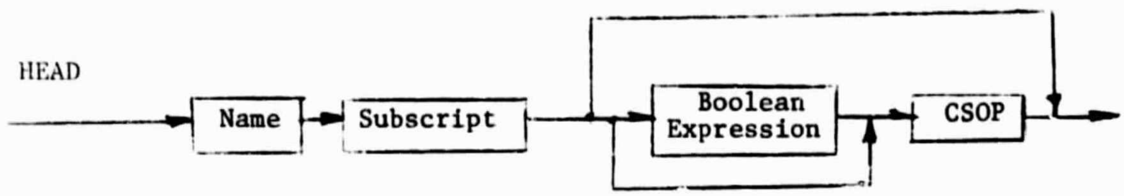
STATE



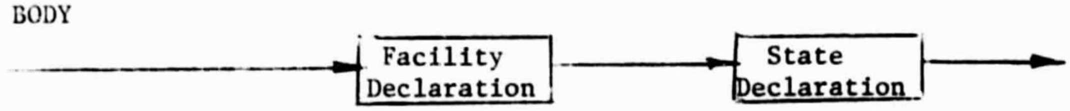
AUTOMATION



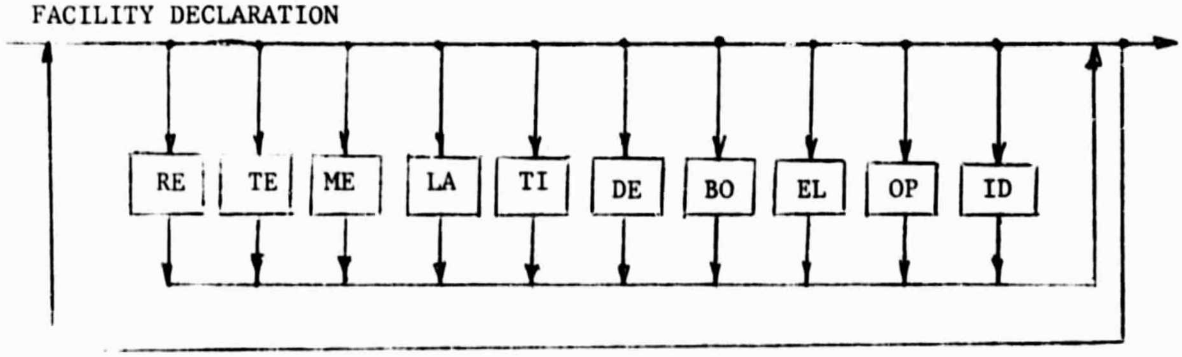
HEAD



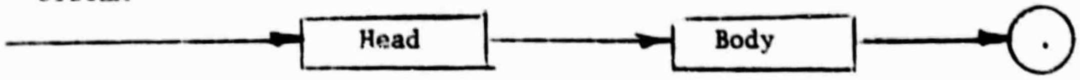
BODY



FACILITY DECLARATION

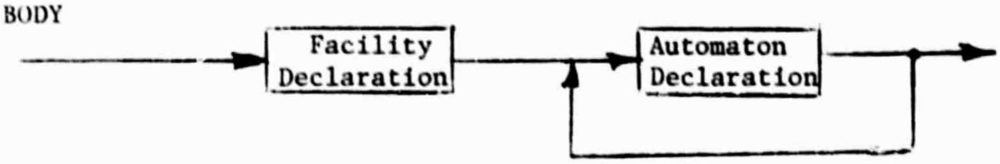


SYSTEM

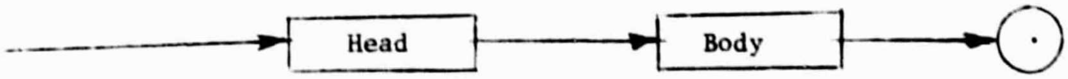


HEAD - Same as Automaton

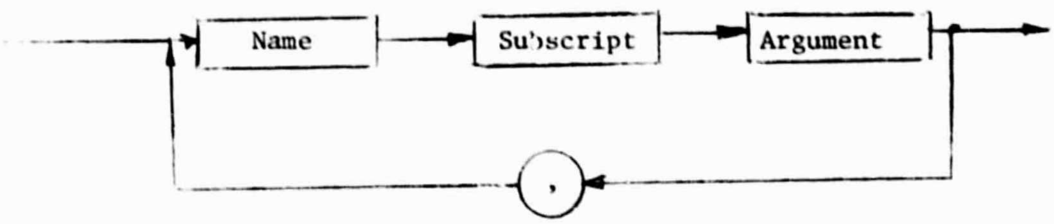
BODY



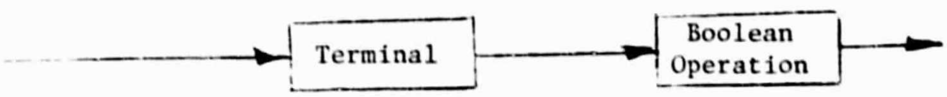
OPerator



HEAD



BODY

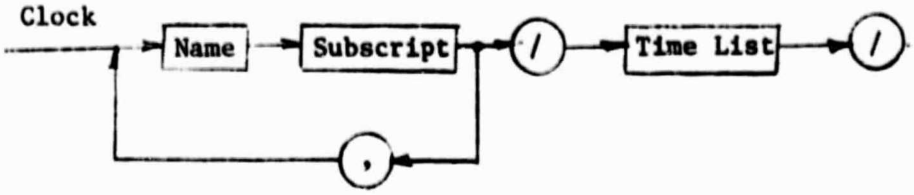


APPENDIX B. SYNTAX DIAGRAMS FOR SIMULATOR COMMANDS

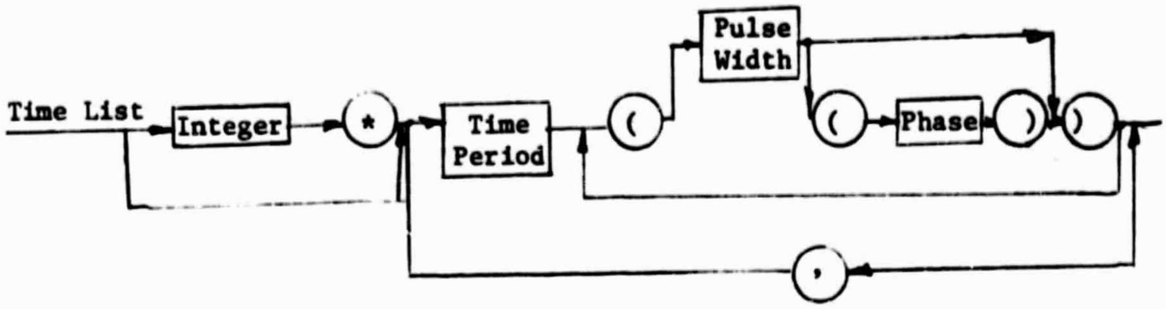
APPENDIX B. SYNTAX DIAGRAMS FOR SIMULATOR COMMANDS

The syntax diagrams for the DDL simulator commands are given in this appendix. For the detailed description and exampled of DDL simulator commands refer to [6].

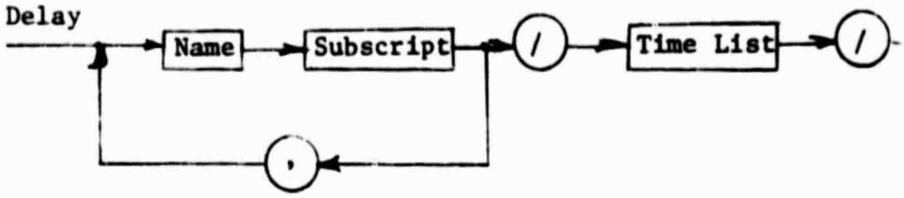
Clock



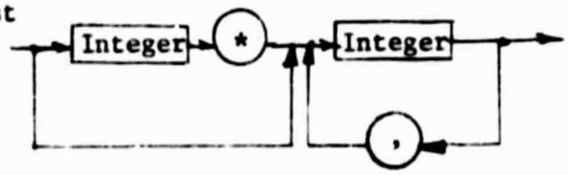
Time List



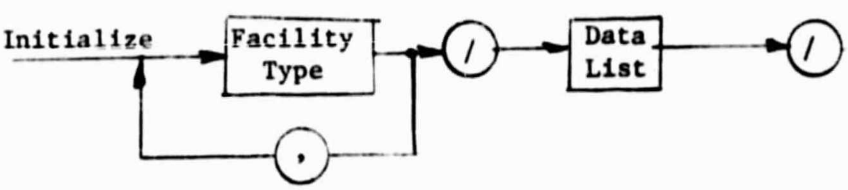
Delay



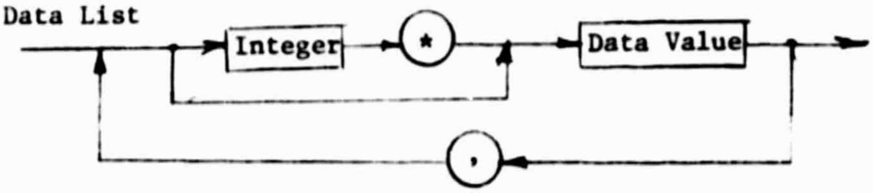
Time List
(Delay)

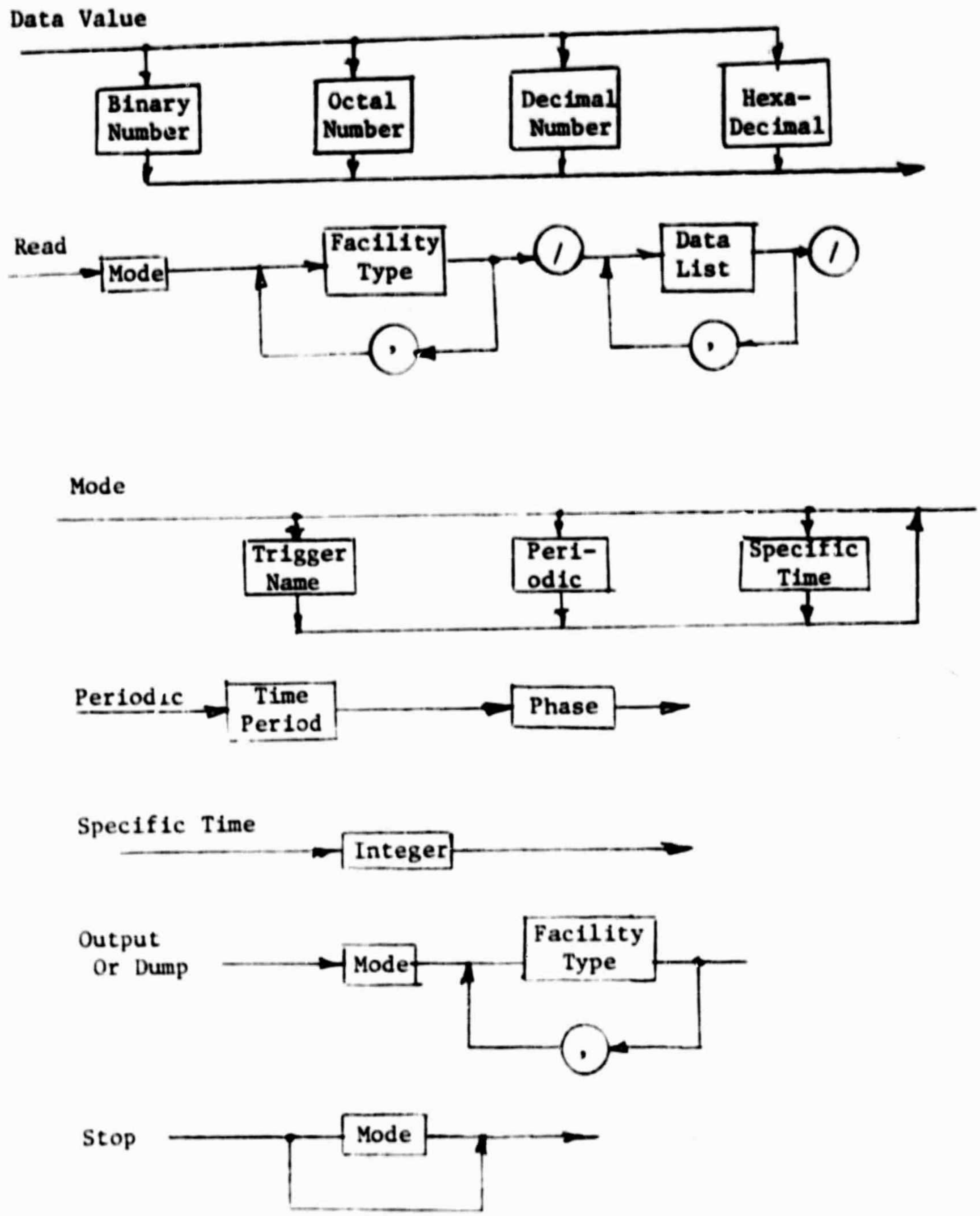


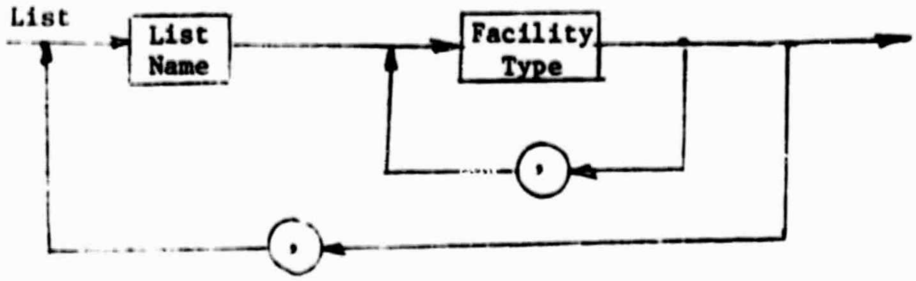
Initialize

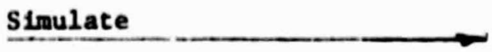


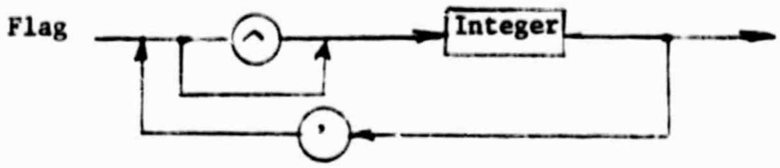
Data List

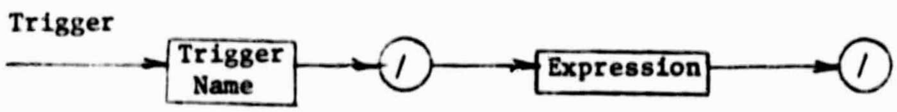


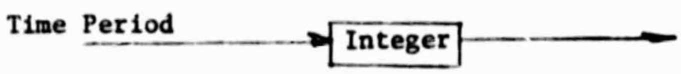

Data Value
Binary Number
Octal Number
Decimal Number
Hexa-Decimal
Read
Mode
Facility Type
/
Data List
/
,
,
Mode
Trigger Name
Peri-odic
Specific Time
Periodic
Time Period
Phase
Specific Time
Integer
Output Or Dump
Mode
Facility Type
,
Stop
Mode

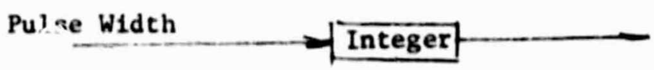

List
List Name
Facility Type
,
,

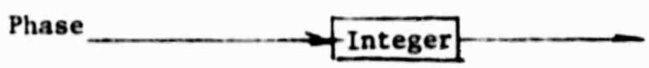

Simulate


Flag
^
Integer
,


Trigger
Trigger Name
/
Expression
/


Time Period
Integer

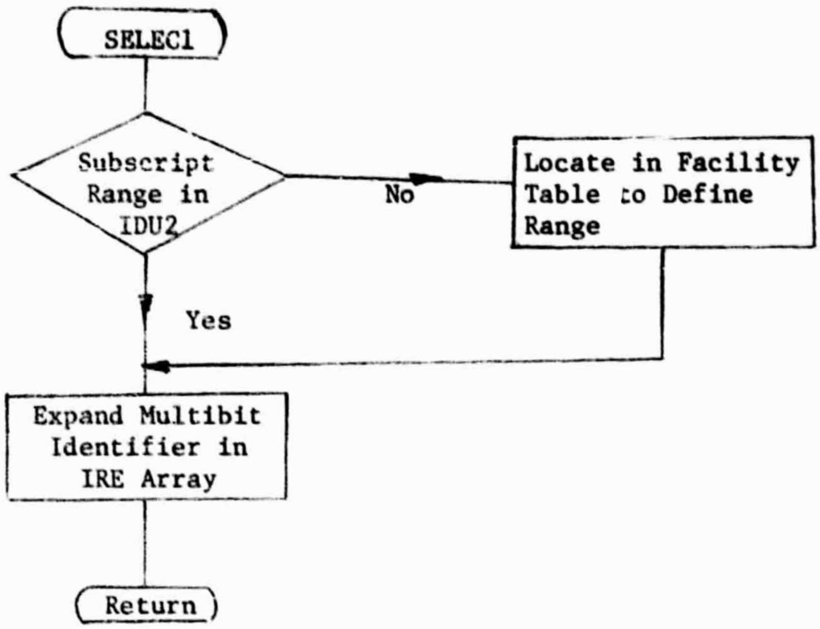

Pulse Width
Integer

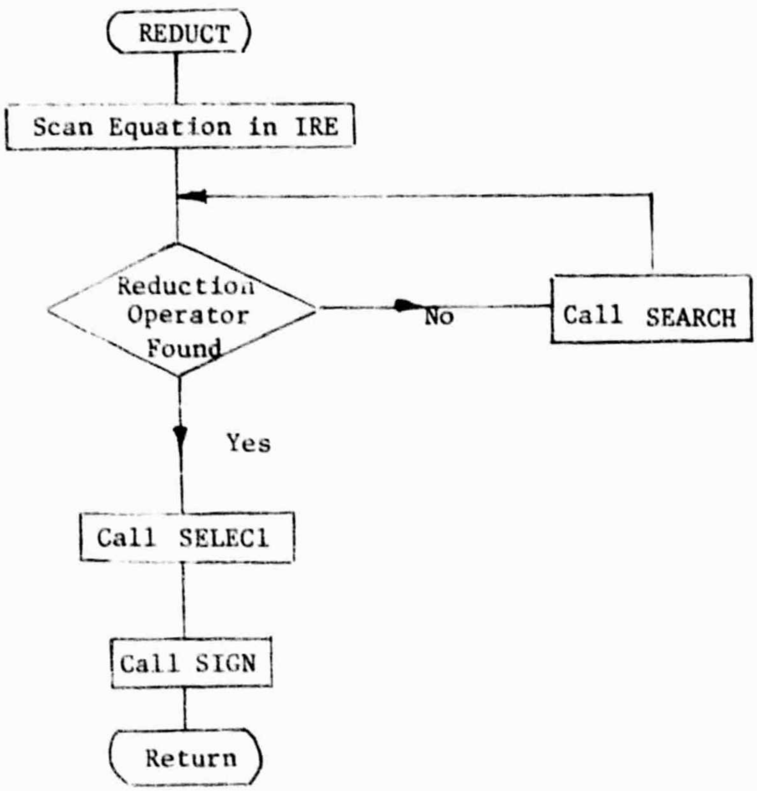

Phase
Integer

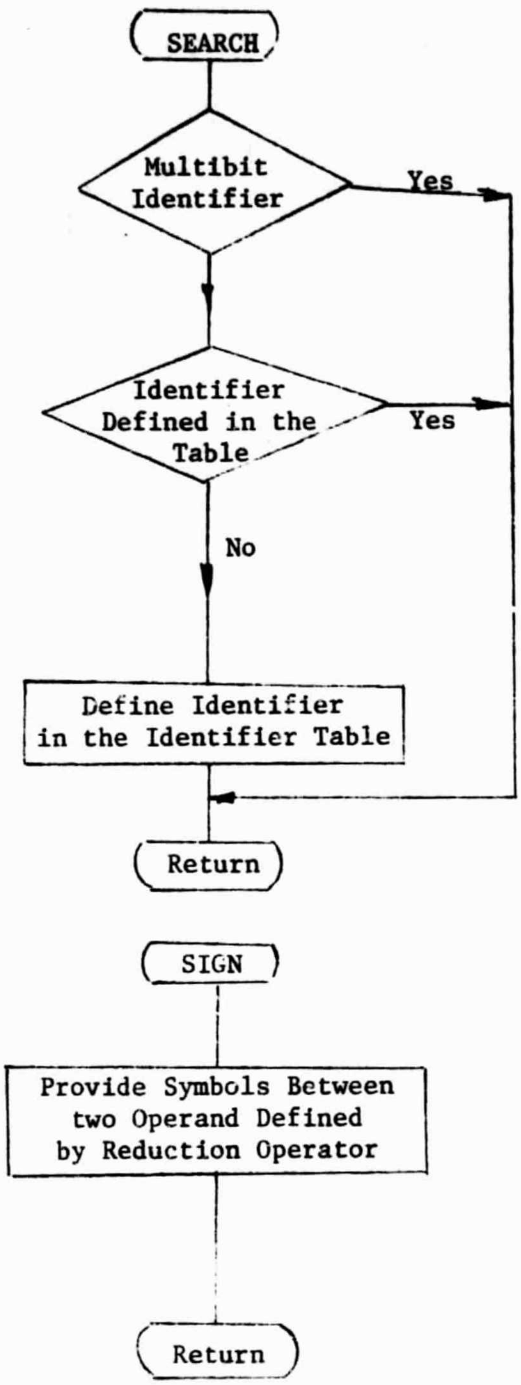
APPENDIX C. FLOWCHARTS

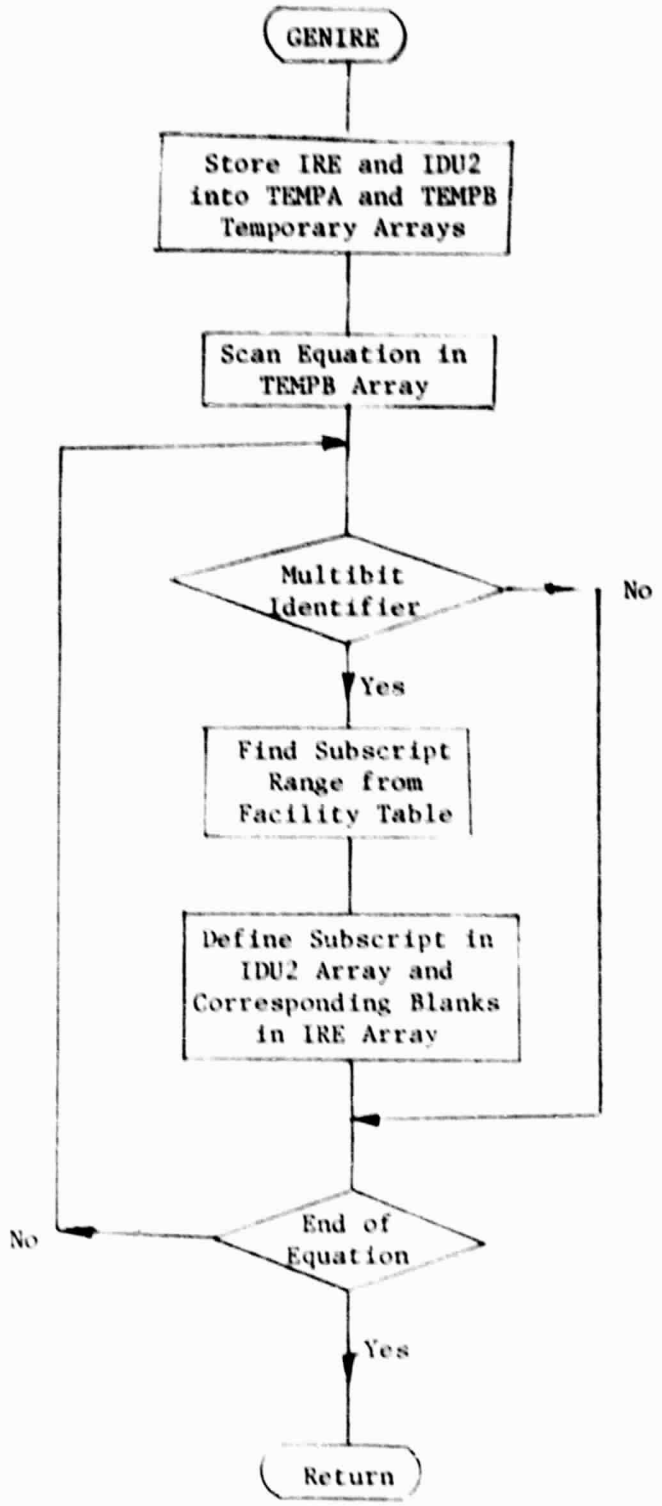
APPENDIX C. FLOWCHARTS

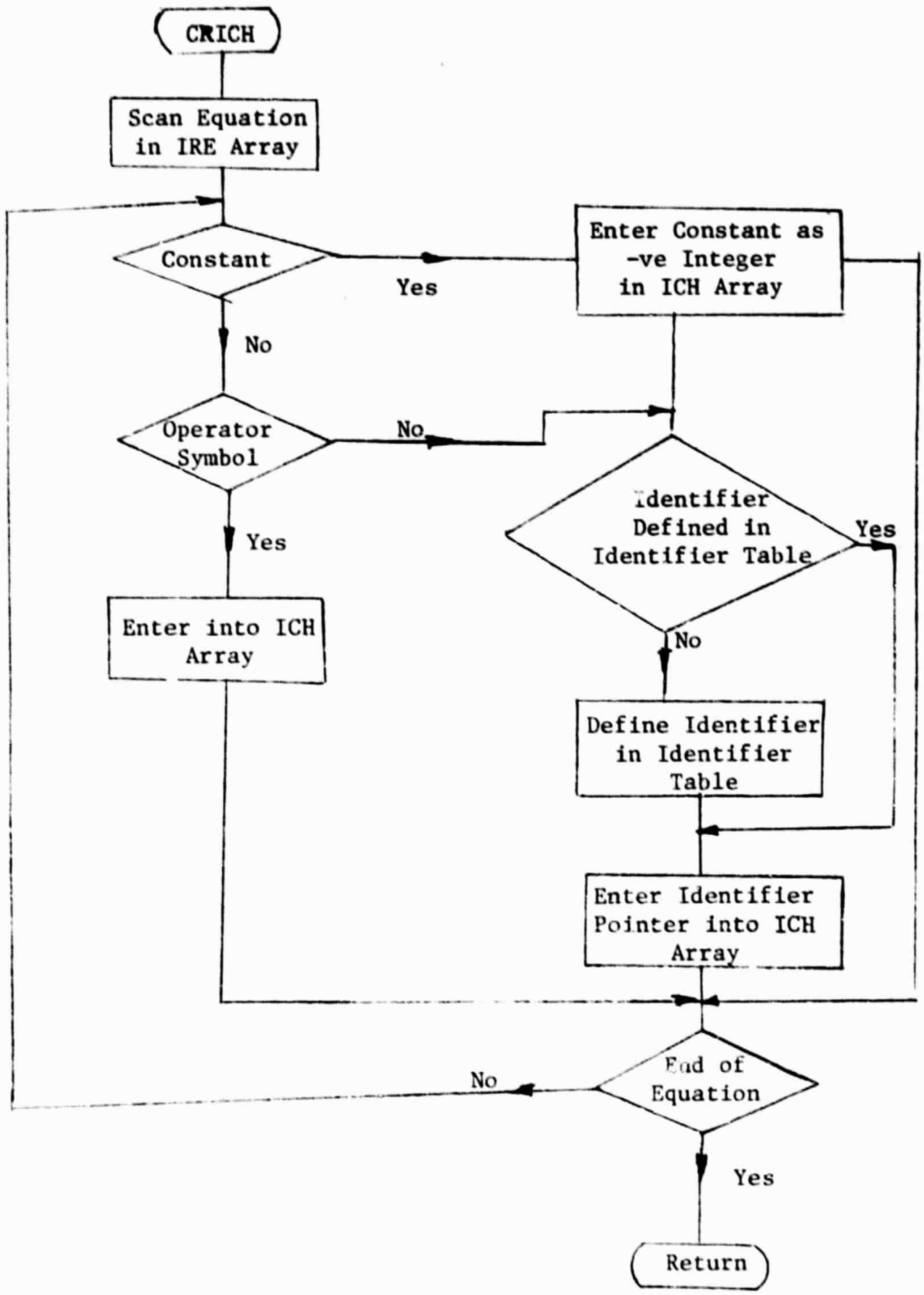
This appendix presents the detailed flowcharts of the synthesis program. Refer to Table 6 and 7 in Chapter 5 for the array names and variables used in the flowcharts.

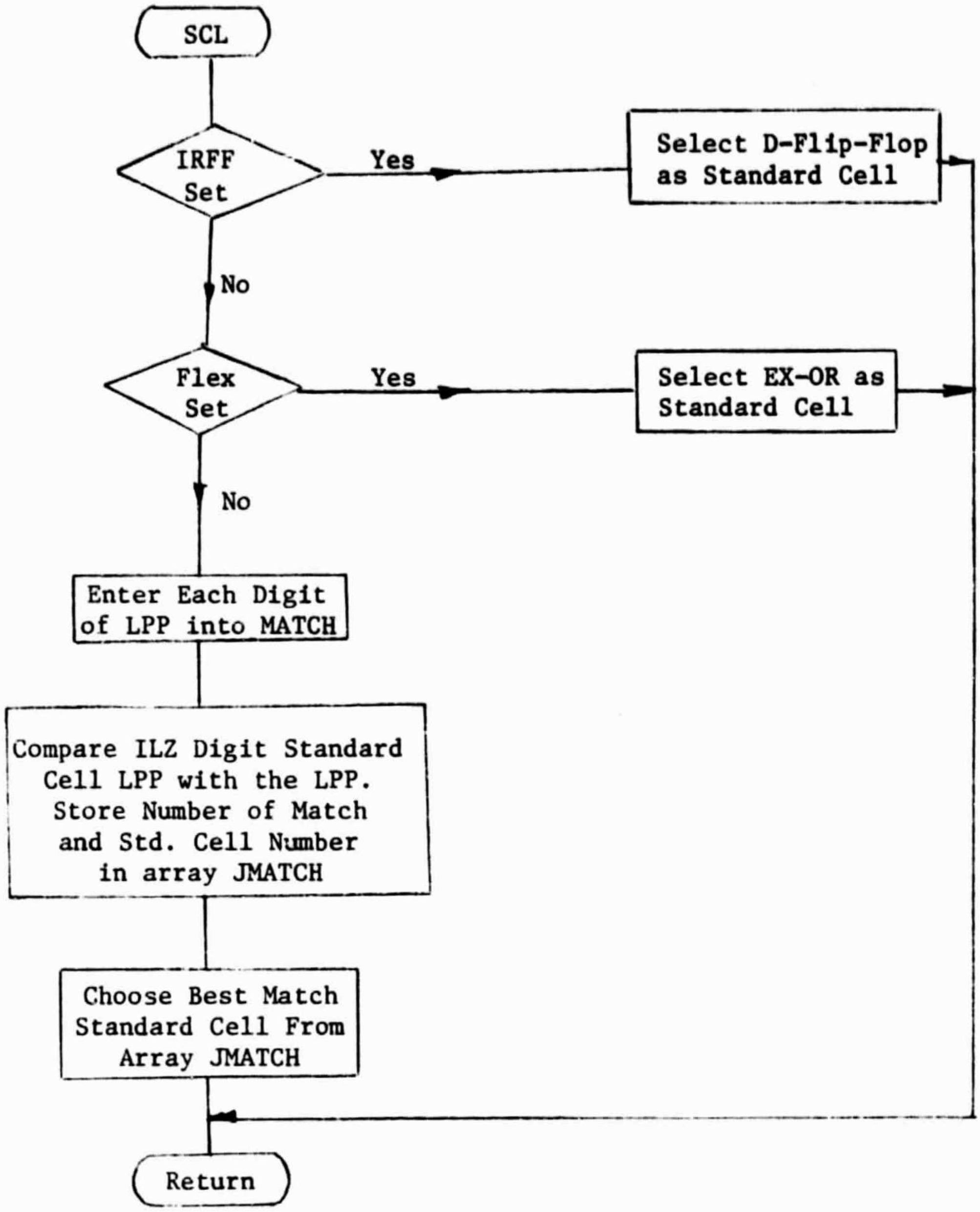

SELEC1
Subscript Range in IDU2
No
Locate in Facility Table to Define Range
Yes
Expand Multibit Identifier in IRE Array
Return

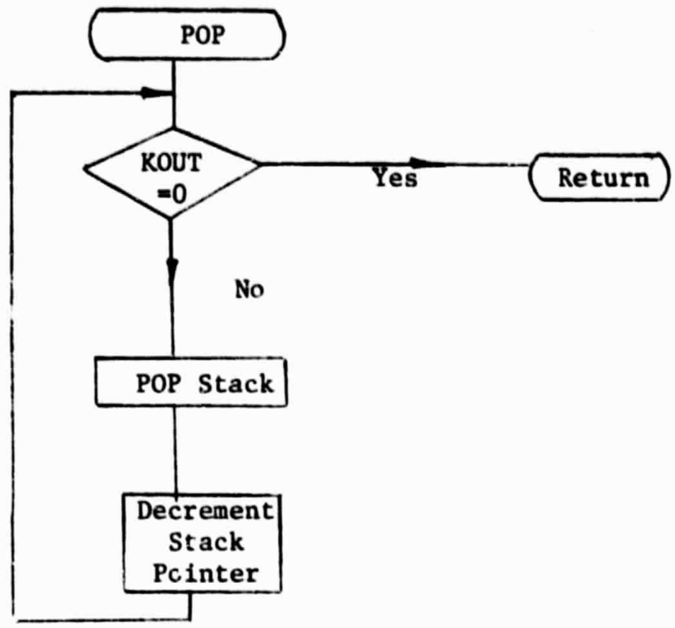

REDUCT
Scan Equation in IRE
Reduction Operator Found
No
Call SEARCH
Yes
Call SELEC1
Call SIGN
Return

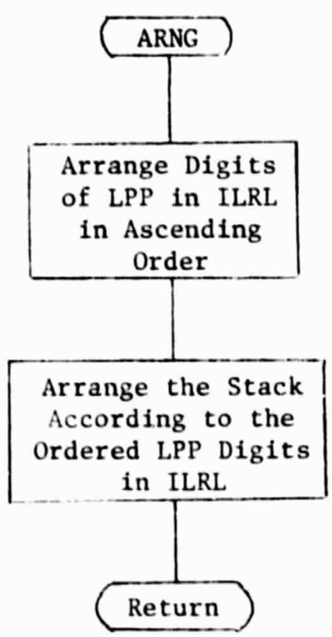

SEARCH
Multibit Identifier
Yes
Identifier Defined in the Table
Yes
No
Define Identifier in the Identifier Table
Return
SIGN
Provide Symbols Between two Operand Defined by Reduction Operator
Return

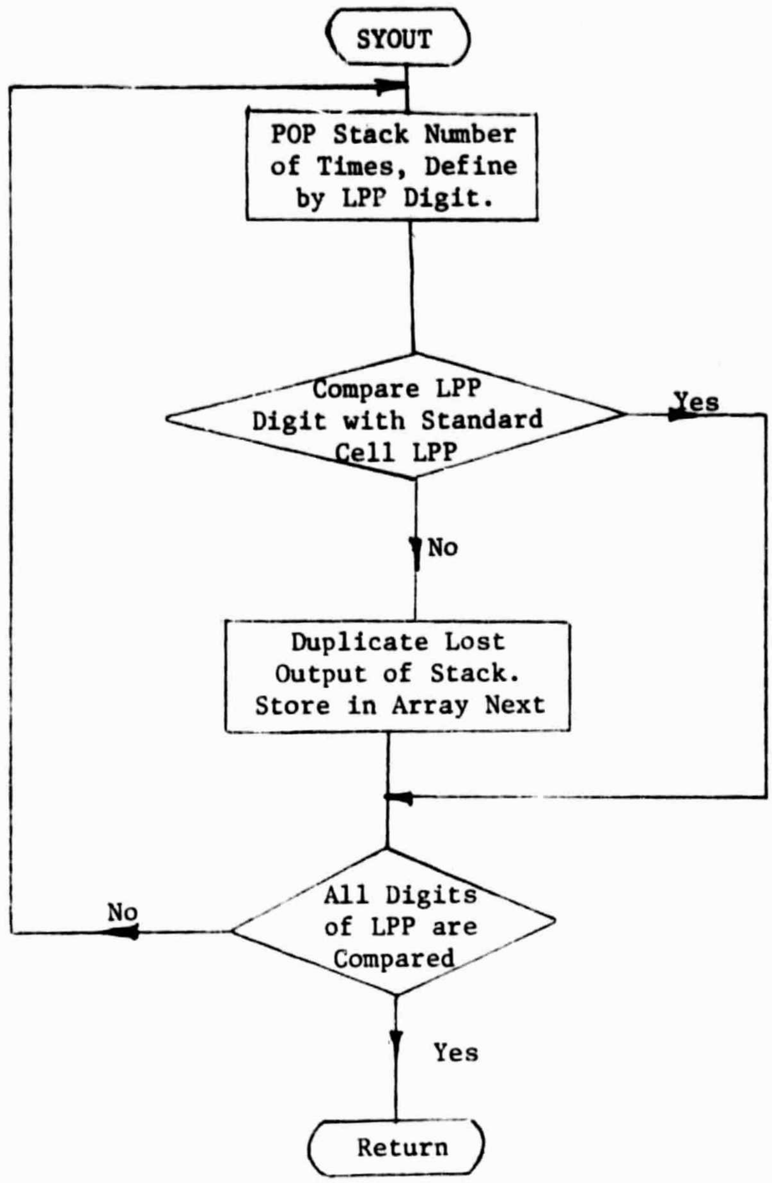

GENIRE
Store IRE and IDU2 into TEMPA and TEMPB Temporary Arrays
Scan Equation in TEMPB Array
Multibit Identifier
No
Yes
Find Subscript Range from Facility Table
Define Subscript in IDU2 Array and Corresponding Blanks in IRE Array
End of Equation
No
Yes
Return

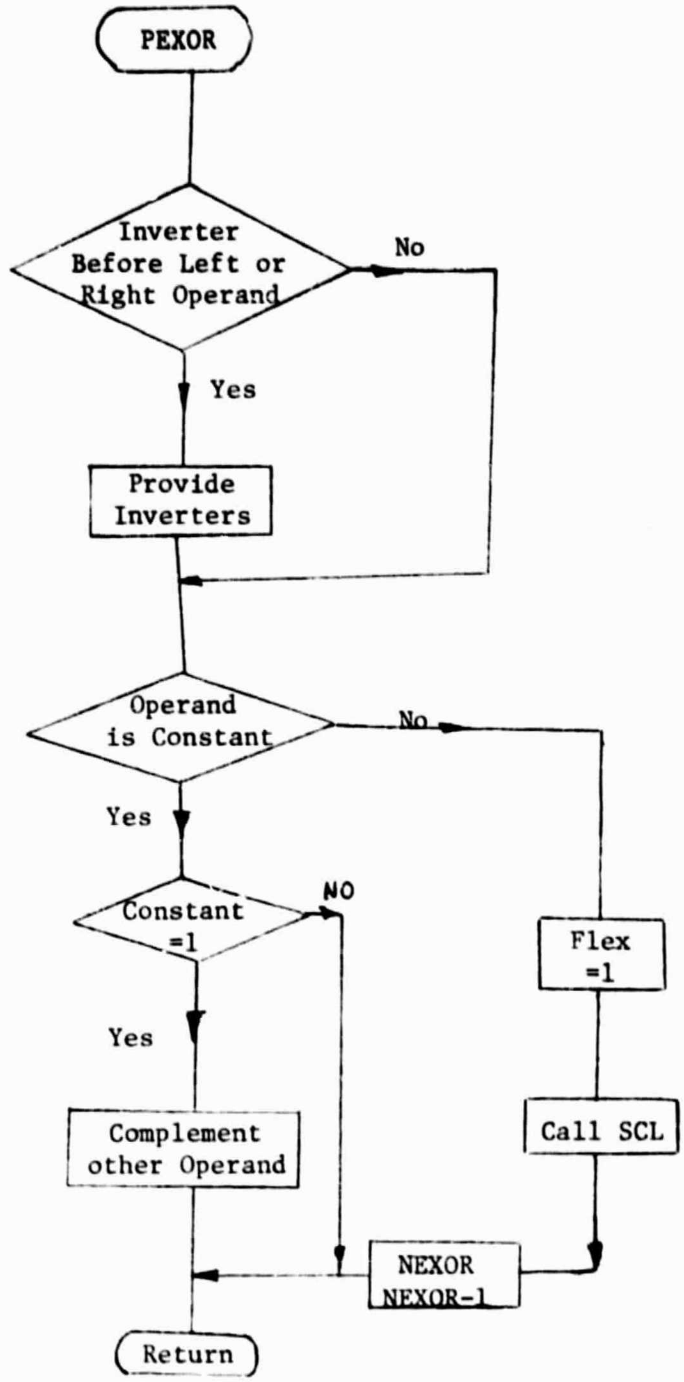

CRICH
Scan Equation in IRE Array
Constant
Yes
Enter Constant as -ve Integer in ICH Array
No
Operator Symbol
No
Identifier Defined in Identifier Table
Yes
Yes
Enter into ICH Array
No
Define Identifier in Identifier Table
Enter Identifier Pointer into ICH Array
End of Equation
No
Yes
Return

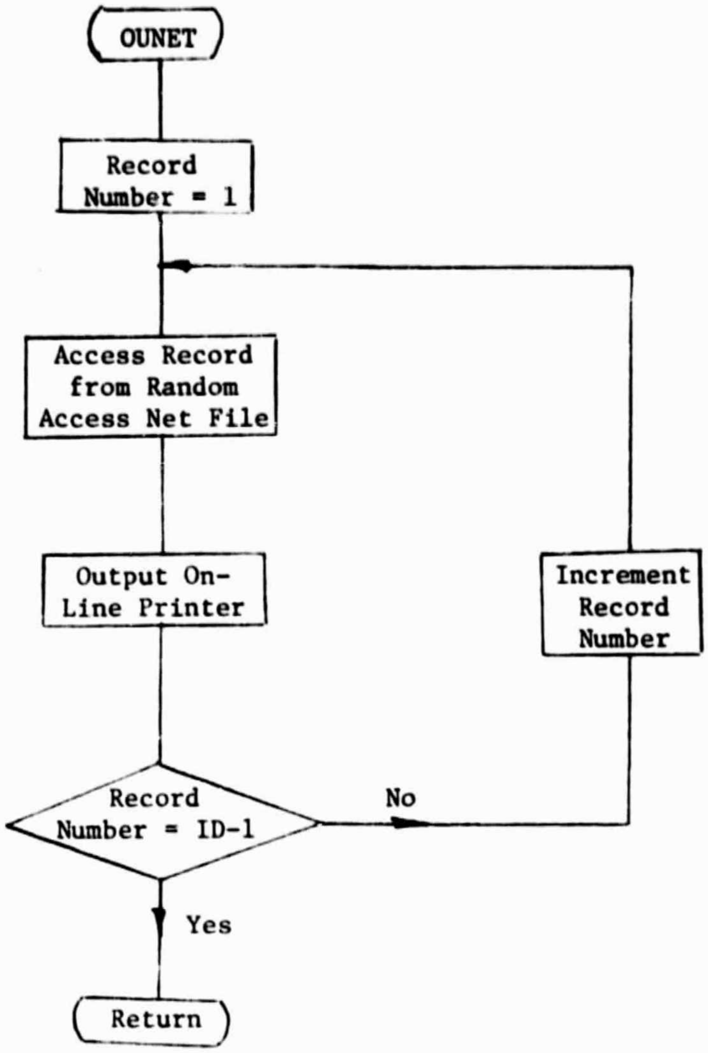

SCL
IRFF Set
Yes
Select D-Flip-Flop as Standard Cell
No
Flex Set
Yes
Select EX-OR as Standard Cell
No
Enter Each Digit of LPP into MATCH
Compare ILZ Digit Standard Cell LPP with the LPP. Store Number of Match and Std. Cell Number in array JMATCH
Choose Best Match Standard Cell From Array JMATCH
Return

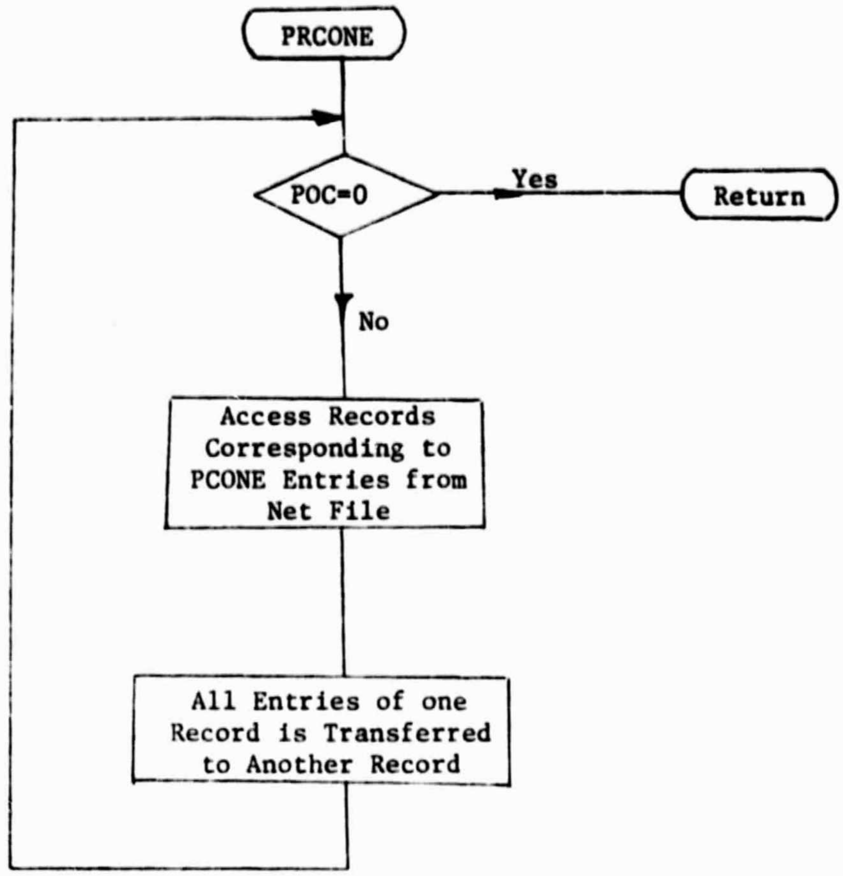

POP
KOUT
=0
Yes
Return
No
POP Stack
Decrement
Stack
Pcinter


ARNG
Arrange Digits
of LPP in ILRL
in Ascending
Order
Arrange the Stack
According to the
Ordered LPP Digits
in ILRL
Return

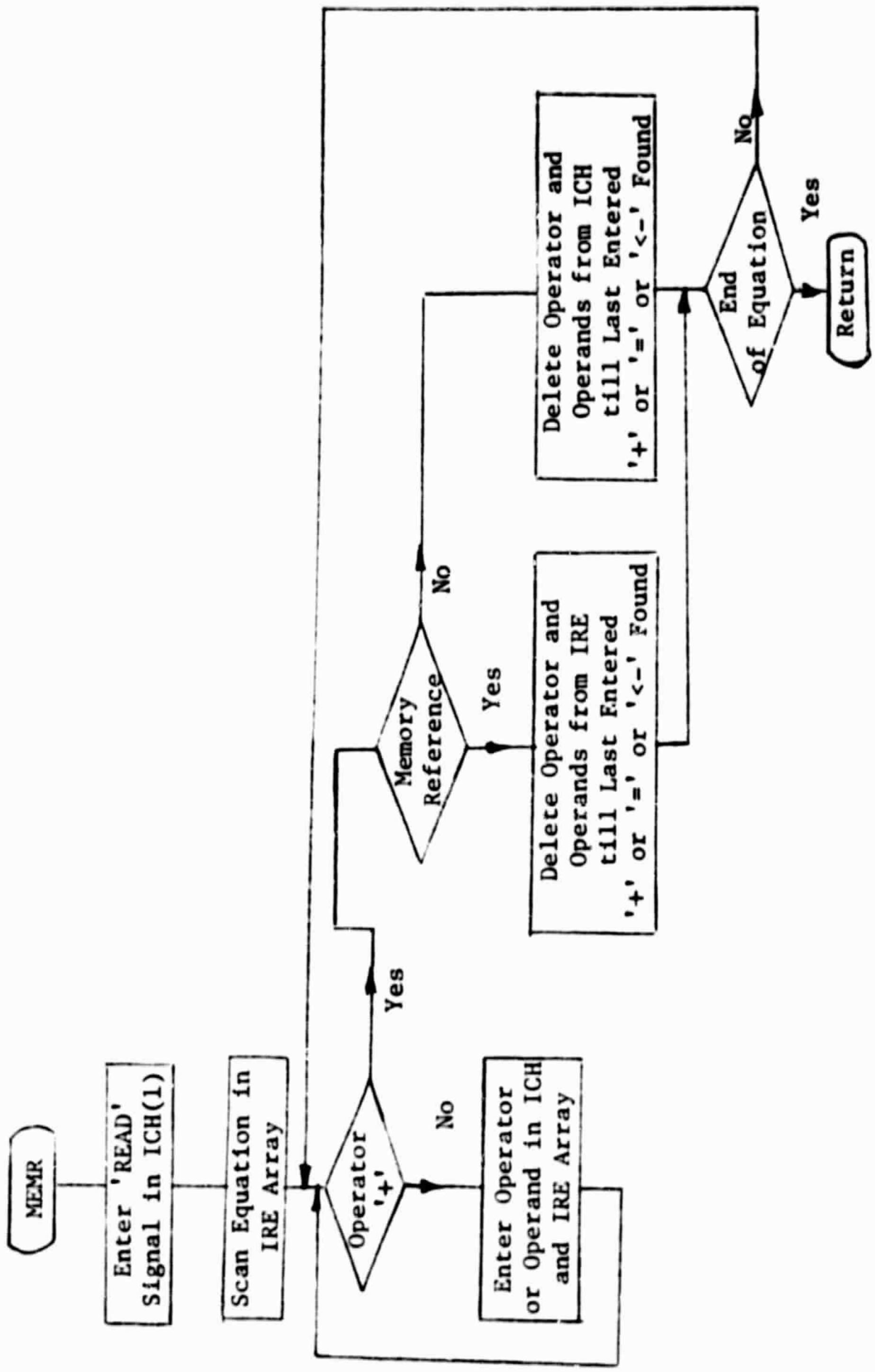

SYOUT
POP Stack Number of Times, Define by LPP Digit.
Compare LPP Digit with Standard Cell LPP
Yes
No
Duplicate Lost Output of Stack. Store in Array Next
All Digits of LPP are Compared
No
Yes
Return

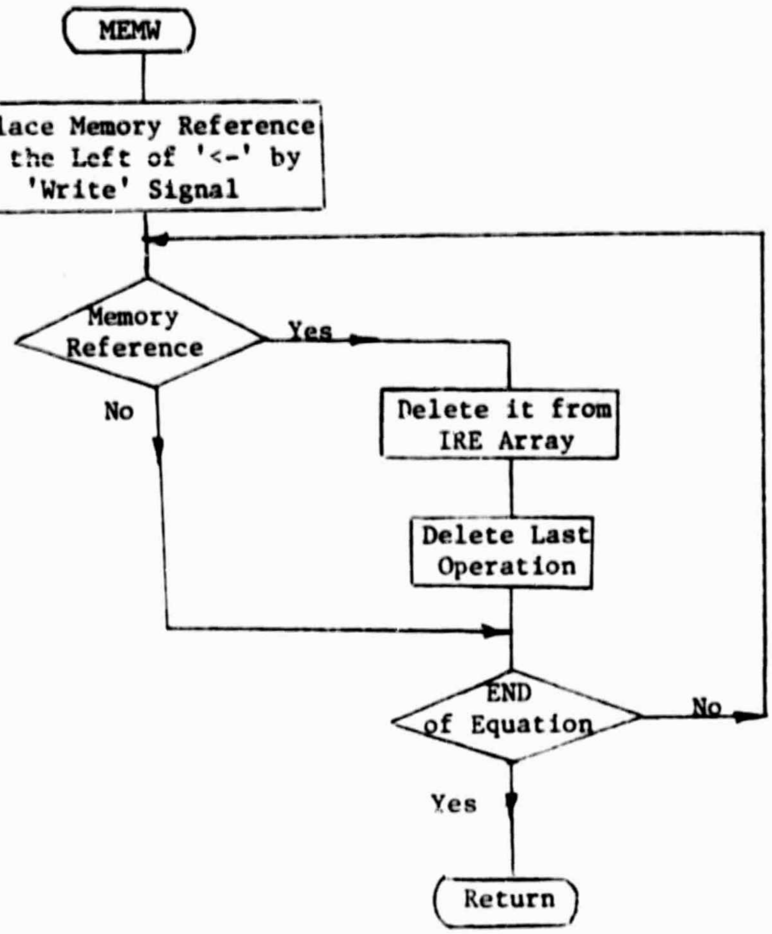

PEXOR
Inverter Before Left or Right Operand
No
Yes
Provide Inverters
Operand is Constant
No
Yes
Constant =1
NO
Yes
Complement other Operand
Flex =1
Call SCL
NEXOR NEXOR-1
Return

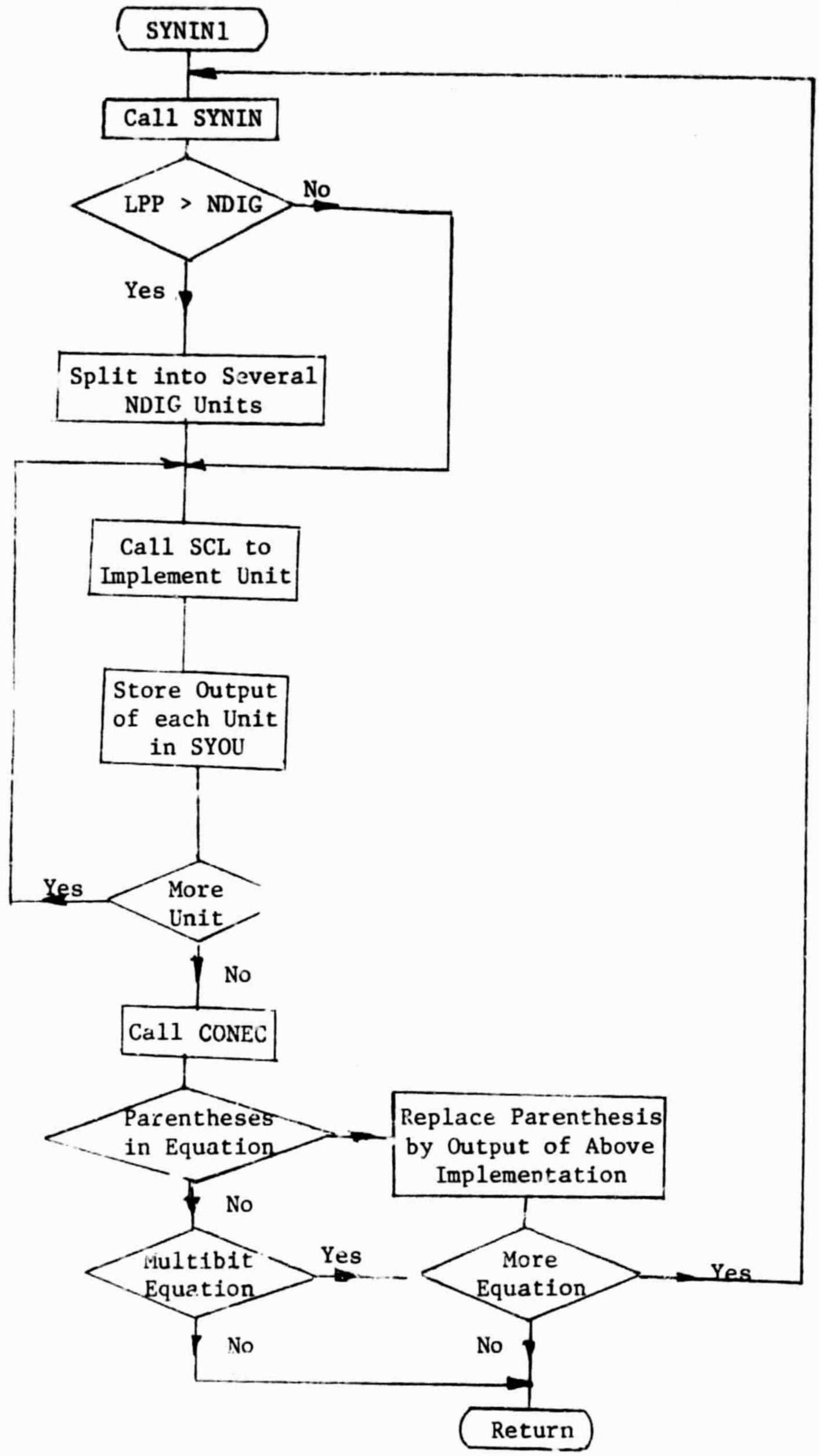

OUNET
Record Number = 1
Access Record from Random Access Net File
Output On-Line Printer
Increment Record Number
Record Number = ID-1
No
Yes
Return

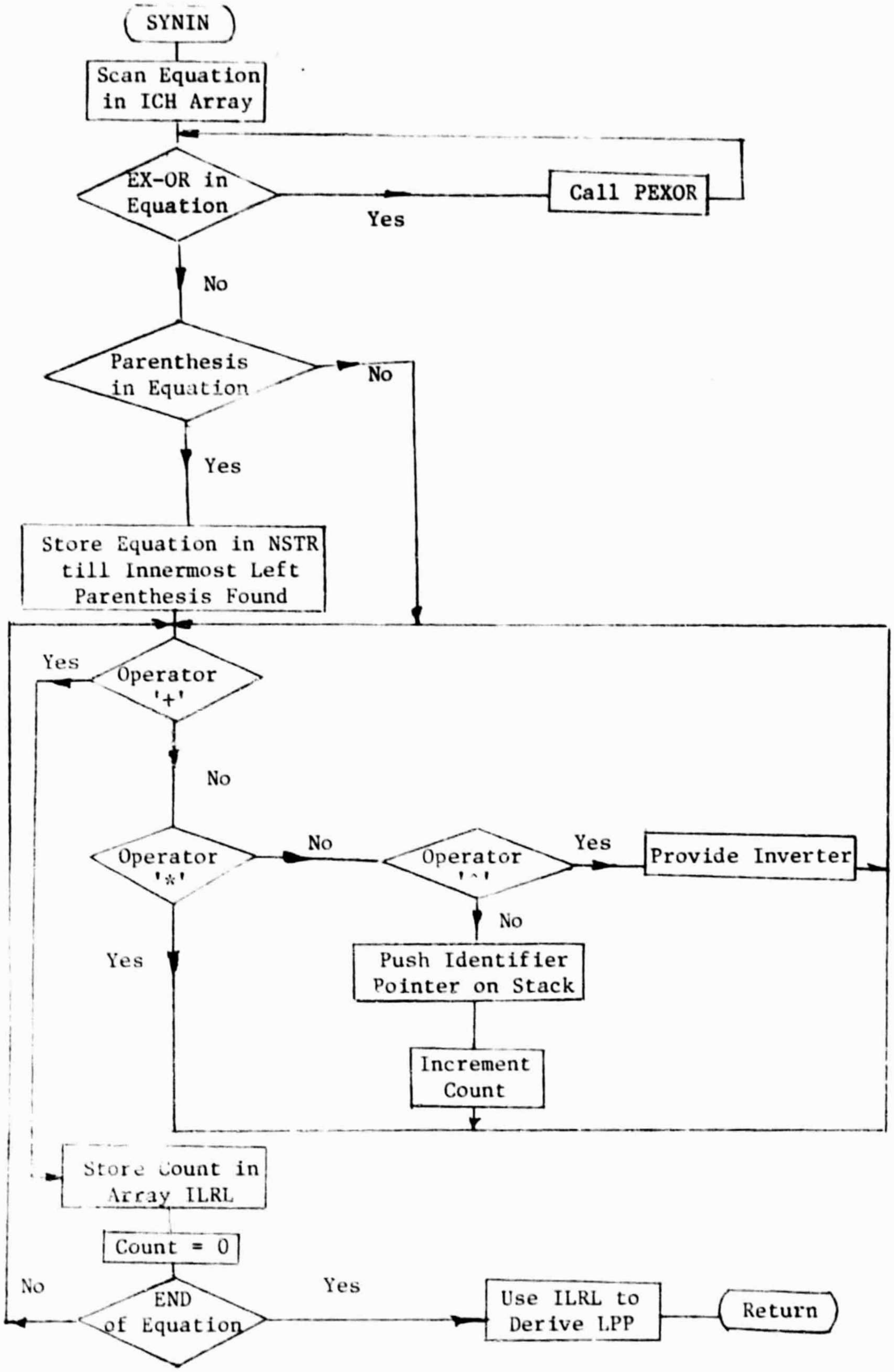


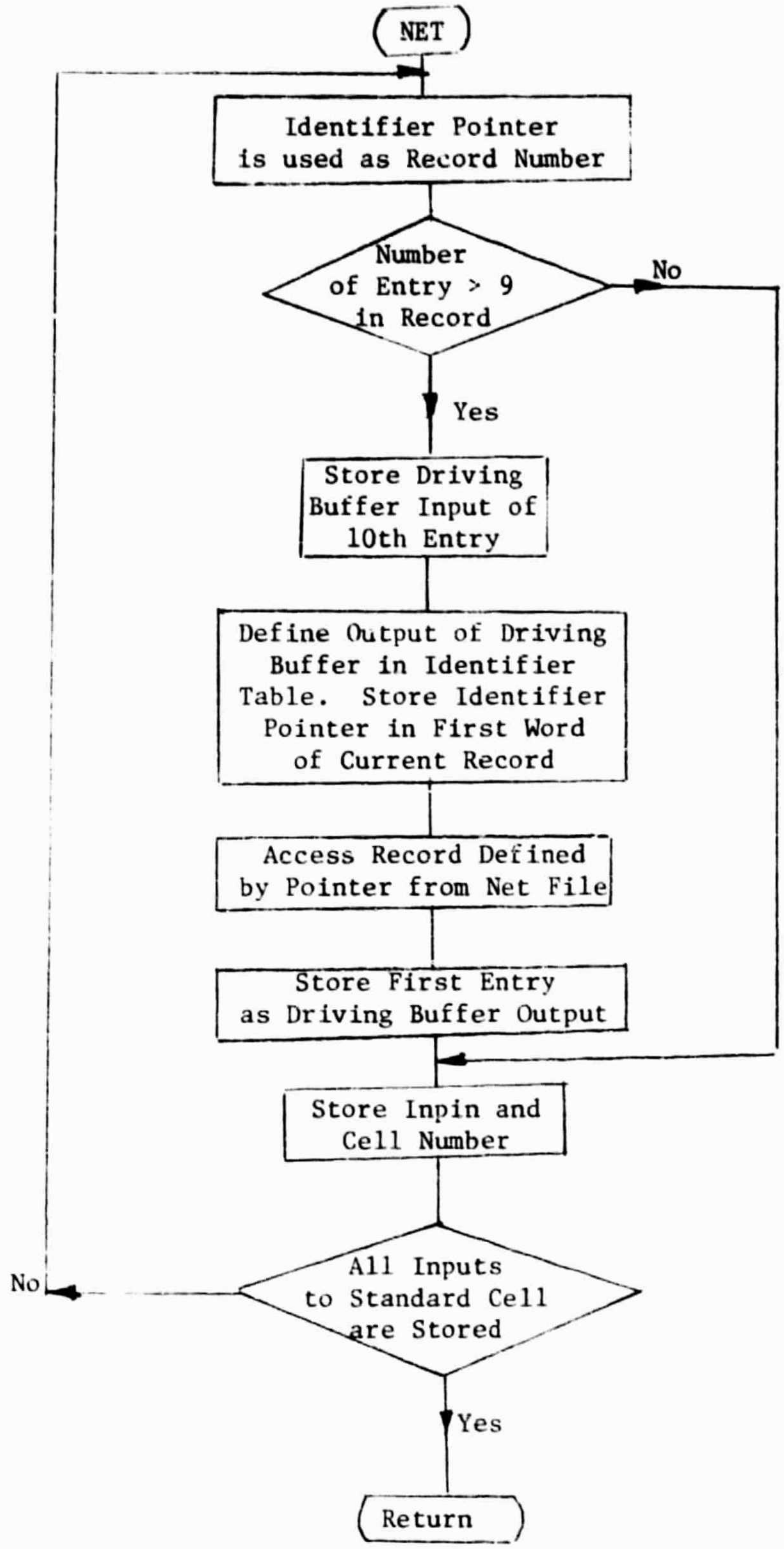
PRCONE
POC=0
Yes
Return
No
Access Records Corresponding to PCONE Entries from Net File
All Entries of one Record is Transferred to Another Record


MEMR
Enter 'READ' Signal in ICH(1)
Scan Equation in IRE Array
Operator '+'
No
Enter Operator or Operand in ICH and IRE Array
Yes
Memory Reference
Yes
Delete Operator and Operands from IRE till Last Entered '+' or '=' or '<-' Found
No
Delete Operator and Operands from ICH till Last Entered '+' or '=' or '<-' Found
End of Equation
No
Yes
Return


MEMW
Replace Memory Reference to the Left of '<-' by 'Write' Signal
Memory Reference
Yes
No
Delete it from IRE Array
Delete Last Operation
END of Equation
No
Yes
Return


SYNIN1
Call SYNIN
LPP > NDIG
No
Yes
Split into Several NDIG Units
Call SCL to Implement Unit
Store Output of each Unit in SYOU
More Unit
Yes
No
Call CONEC
Parentheses in Equation
Replace Parenthesis by Output of Above Implementation
No
Multibit Equation
Yes
More Equation
Yes
No
No
Return


SYNIN
Scan Equation in ICH Array
EX-OR in Equation
Yes
Call PEXOR
No
Parenthesis in Equation
No
Yes
Store Equation in NSTR till Innermost Left Parenthesis Found
Yes
Operator '+'
No
Operator '*'
No
Operator '^'
Yes
Provide Inverter
No
Yes
Push Identifier Pointer on Stack
Increment Count
Store Count in Array ILRL
Count = 0
No
END of Equation
Yes
Use ILRL to Derive LPP
Return


NET
Identifier Pointer is used as Record Number
Number of Entry > 9 in Record
No
Yes
Store Driving Buffer Input of 10th Entry
Define Output of Driving Buffer in Identifier Table. Store Identifier Pointer in First Word of Current Record
Access Record Defined by Pointer from Net File
Store First Entry as Driving Buffer Output
Store Inpin and Cell Number
All Inputs to Standard Cell are Stored
No
Yes
Return